TITLE:
Local Parallel Picture Processing and Hardware Implementation(Dissertation_全文)

AUTHOR(S):
Kidode, Masatsugu

CITATION:
Kidode, Masatsugu. Local Parallel Picture Processing and Hardware Implementation. 京都大学, 1980, 工学博士

ISSUE DATE:
1980-01-23

URL:
https://doi.org/10.14989/doctor.r4056

RIGHT:

京都大学学術情報リポジトリ
KURENAI 紅
Kyoto University Research Information Repository

LOCAL PARALLEL PICTURE PROCESSING AND HARDWARE IMPLEMENTATION

MASATSUGU KIDODE

MAY 1979

LOCAL PARALLEL PICTURE PROCESSING AND HARDWARE IMPLEMENTATION

MASATSUGU KIDODE

MAY 1979

DOC
1979
20
電気系

LOCAL PARALLEL PICTURE PROCESSING AND HARDWARE IMPLEMENTATION

MASATSUGU KIDODE

ABSTRACT

This thesis describes research on the development of local parallel picture processing techniques for edge detection and texture analysis, and the hardware implementation of local parallel picture processing functions.

Local parallel operations are successfully used for detecting edge elements from photographs of human face and for analyzing statistical properties of texture patterns. Several local operators are designed to demonstrate a wide applicability of local parallel picture processing.

The implemented hardware consists of a microprogrammable controller, scratch pad memory, address controller and several special-purpose functional modules. It is designed to accomplish some basic functions at high speed, so that it can realize picture processing in a better cost-effectiveness.

Experimental results indicate that this hardware is powerful enough to explore many different kinds of applications and their quick analysis in the picture processing studies.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to Professor Toshiyuki Sakai for his insights, academic advice and encouragement.

I am also grateful to Professor Makoto Nagao for his personal guidance.

It is my pleasure to make special notes of the following individuals who have significantly influenced my career and research study: Dr. Hiroji Nishino of ElectroTechnical Laboratory for his direction and support, Prof. King-Sun Fu of Purdue University for his academic guidance, Dr. Harry Wechsler for his collaboration on edge detection and texture analysis, Dr. Kenji Kakizaki for his critical advice and for facilitating my research study, Drs. Tadaaki Tarui and Hiroshi Genchi for their arrangements to complete my research, and Drs. Ken-ichi Mori and Sadakazu Watanabe for their direct guidance and encouragement.

Other people who so kindly extended their assistance and cooperation in my preparation of this thesis are: Dr. Takeo Kanade, Hidenori Shinoda, Haruo Asada, Hideyuki Tamura, Takashi Matsuyama, and my colleagues. In particular, Hidenori Shinoda and Haruo Asada contributed significantly to the technical content of Chapters 3 and 4.

Above all, the love and support of my wife, Toshiko, contributed immensly and indisputably to the completion of this thesis. I will remain deeply in debt to her.

May 1979
Masatsugu KIDODE

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

Picture processing by digital computers is a current topic in the field of research on medical diagnosis, remote sensing and industrial automation. The principal stimulus for the computer applications to these topics is their capability to perform much more complex picture processing functions than by analog methods. With the computer's powerful ability of achieving radiometric and geometric transformations, manipulating pictures and extracting new kinds of information, a picture analyst such as a radiologist, a meteorologist, a geologist, even a worker has been continuing to study more complex and challenging problems in digital picture processing.

This thesis is concerned with picture processing of this kind, which are, roughly speaking, as follows:

(1) Local Parallel Picture Processing by Computer - New Techniques

Local parallel operations are successfully used for detecting edge elements from photographs of human face and for analyzing statistical property of texture patterns. Some examples with variation of size and operation type in the local area indicate a wide and potential applicability of the local parallel picture processing for pattern recognition problems.

(2) Local Parallel Picture Processor - New Aspect

A microprogrammable local parallel picture processor has been implemented in a simple hardware. The processor has several features in its design concepts in order to improve the cost-effectiveness of picture processing. It has been designed to perform such basic picture processing functions as two dimensional convolution,

affine coordinate transformation, logical filtering, region labeling, data conversion, historgram computation and pixelwise operations.

Experimental results demonstrate that the processor is powerful and economical enough to explore novel applications and quick analysis in the picture processing studies.

1-1 Software Implementation of Local Parallel Picture Processing

By a "local parallel" operation (or processing) is meant a function whose values at any given point depend only on a small set of its neighbors (local area) in a given input picture and whose values do not depend their values at any other points[1][2][3] from the point of view in software implementation, and/or an operation which is carried out locally in parallel: i.e., parallel data access and parallel process in the local area from the point of view in hardware implementation. The value obtained by a local operation may take a value of the corresponding point in the transformed output picture or a value of the extracted feature.

Many useful local parallel operations have been found to transform a given picture into another one and extract a set of features from a given picture[4][5]. These operations are applied not only to gray scale pictures but also to binary pictures.

For example, a contour picture shown in Fig. 1-2 can be detected from an original picture in Fig. 1-1 by an edge detector which computes two dimensional differentiation. Similarly, application of a thinning operator on the picture in Fig. 1-2 which contains thick lines, results in Fig. 1-3.

These two processings are achieved by performing local parallel operations on input pictures. The former local operation

Figure I-1 Original Picture (Human Face)

Figure I-2 Extracted Contour Picture by Laplacian

Figure I-3 Thinned Contour Picture

is a derivative operation on the gray scale picture, which gives high values at points where the gray levels change abruptly. Many operators of this type can be used as an edge detector whose values indicate edge strengths, and then edge points are detected by thresholding these values.

In the latter thinning local operation for the binary picture, the basic idea is to delete border points which have more than one neighboring point. The thinning operation will be iterated until no deletion of points is possible.

Here arise the following questions in performing local parallel operations on a given picture:

(1) Size of Local Area

How can we determine the size of the local area (neighborhood) ? On what neighboring points does the value at a point depend ?

(2) Shape of Local Area

How can we choose the shape of the local area in terms of orientation or directionality ? Is an operation sensitive to a particular direction or not ?

(3) Complexity of Local Operation

What kind of computations are taken in the local area ?

Simpler operations are: additions or subtractions, logical operations, sum-of-products, series expansion etc.

(4) Program Control Structure

In what manner are local operations applied on a picture ? Are they local operations by scanning the picture horizontally and then vertically, or vice versa ? Is a tracing algorithm necessary ?

These problems strongly influence implementation of the operations by digital computers in terms of program control, data access control and computing process.

From the technical point of view in software implementation, we should study the availability of local operations to digital picture processing, and then implement them as simply as possible, that is, as efficiently as possible they can be.

In this thesis, new local parallel picture processing techniques are successfully implemented in edge detection and texture analysis problems of current concern. Two independent edge detection procedures of differentiation type are taken, while a new planar random walk approach is realized to calculate texture parameters by the local processing.

Related Works

Many papers have been published on edge detection and texture analysis. Some selective works which are closely related to the content of this thesis are referred to here.

Nagao and Kanade[6], and Davis[7] give excellent surveys on edge detection techniques as well as characterizations of the

different algorithms.

An edge element represents the boundary between two adjacent regions with a small neighborhood of picture elements, assuming that the regions differ markedly with respect to at least one feature, for example, gray level. The decision on whether or not a picture element is on an edge is made on the basis of the gray level of its neighbors (local area) by using differential operations, statistical methods, optimal approaches, and so on. From the technical points of view in software implementation, the size and shape of local area, decision complexity and program structure are discussed in the following works.

Robinson[8] made use of eight 3x3 compass gradient masks with five level integer weights between -2 and +2. Her gradient masks are an extension of the Sobel operator[9]. She introduced concepts of the local connectivity of edge directions, edge activity index and local adaptive threshold to improve the edge detection performance. Although the weighting constants are based on intuitive grounds, her mask operator is simple and fast[10]. A number of edge detection techniques are based on such local spatial differentiation operators with different weighting values.

Frei and Cheng[11] also examined 3x3 mask operators, whose weights are derived from a set of orthogonal functions related to distinctive image features; they proposed an improved decision rule to extract edges and ways to implement it economically.

These small local operators will be sensitive to noise and/or blurring. Instead of operating on small local areas, one must take account of weighted averaging over local areas. Here how to choose local area size and shape presents a problem. A large operator will average out the noise, but will respond at multiple locations near the edge. If the operator size is large enough to cover a non-homogeneous region, then the averaging is no longer valid. The size and shape are strongly dependent on spatial frequency response of

desirable objects in the picture.

Rosenfeld and Thurston[12] employed a family of differential operators of every size. This method computes products of the differences between the average gray levels for pairs of neighborhoods of all sizes. The edge detector finds the size of the neighborhood which will yield a local maximum. Based upon the size of the neighborhood and the surrounding local maxima, a decision is made as to the existence of an edge and its sharpness. Their method is, however, basically a horizontal and vertical edge detection technique which may be susceptible to the directions of the edge.

Another approach to edge detection is based on statistial principle. Chow and Kaneko[13] developed a threshold method to detect boundaries in radiographic pictures. They characterized a local area of the picture containing a portion of the boundary as a statistical model with a mixture of two normal intensity distributions. Then, thresholds were determined dynamically according to the local area characteristics. The method is fairly insensitive to shading or gradually changing interference. Difficulties are still found in the size selection of local area and in the distribution assumption.

Hueckel's edge operator[14] was derived on the basis of a formal edge model. Hueckel attempted to adjust the parameters of a simple edge model to best fit the picture function in a moderately large, roughly circular local area: for example, 7x7. Hueckel's operator has been in wide use and its performance was considered to be better than that of other edge detectors especially for curved objects. However, the required computation is considerably larger than other operations because of his complex computation. A recent paper by Nevatia[15] shows that some experiments to reduce the complexity of computation can result in a significant loss in performance for noisy pictures.

Shirai[16] introduced an artificial intelligence technique, a heterachical program, to transform a given picture of polygons

into line drawings. He used only one size (3x3) edge detector and attempted to find objects without a rigid ordering of steps and to use previous results as analysis proceeds. His method is based on the strategy of recognizing objects step by step, each time making use of previous results. In general, the edge detection technique of this type needs special knowledge and control structure to understand the picture. No further consideration on this kind of knowledge-based systems is given in this thesis because it is apart from the intention of this thesis: local parallel operation and simpler programming technique.

Two independent local parallel operations for edge detection will be studied in this thesis, which takes account of local area size and weights. The Laplacian operator is combined with averaging in order to reduce the noise, while a set of finite differences with dynamic local thresholding are used to obtain a good resultant edge picture.

Tomita, Shirai and Tsuji[17] gave an excellent review of previous works on texture analysis. Weszka, Dyer and Rosenfeld[18] made a comparative study of texture measures.

Textures may be categorized into at least two classes: statistical and structural textures. Statistical textures in a visual scene can be regarded as defined by a set of statistical parameters, while structural textures may be considered to be defined by primitive elements which occur repeatedly, according to well-defined arrangement rules.

The statistical approach derives a set of local measurements over the space domain or the corresponding frequency domain for a given picture. Features based on these measurements are then used for texture discrimination. The structural approach describes a picture with a set of primitives and their placement rule, assuming that primitives can be a priori defined and easily identified in

the picture. Only the statistical approach will be discussed, which is close to the content of this thesis.

Haralick, Shanmugam and Dinstein[19] proposed a variety of texture measures specified by a set of gray tone spatial dependence matrices which estimate the joint gray level distribution of pairs of neighboring elements. They suggested some useful texture features for homogeneity, contrast, and directionality in the picture. This method is based on the second-order statistics. It requires normalization of the gray level (histogram equalization) and also depends on orientation and size.

Bajcsy[20] made use of texture features derived from the Fourier power spectrum. She analyzed the coarseness and directionality of texture by the radial and angular distributions of values in the power spectrum. However, the discrete Fourier transform has an inherent edge effect, which introduces spurious horizontal and vertical directionality. These texture features are also sensitive to size and orientation.

Deguchi and Morishita[21] proposed a linear estimation model in the signal detection to segment a given picture by texture features. They assumed that a gray level can be estimated by a linear combination of gray levels in the neighborhood and a white noise. They determined the size of neighborhood and weighting coefficients, a set of texture parameters, by the least-square approximation. The computational cost is a big problem in linear operations by weighting masks.

Higher-order statistics by the gray level run lengths was employed by Galloway[22] to classify terrain types. He computed a set of texture features on the gray level run lengths for representing the coarseness, directionality, and so on. His method requires the gray scale normalization procedure to make all the pictures have a uniform gray level frequency distribution. A comparative study reported that this method is very sensitive to noise[18].

A recent investigation by Tamura, Mori and Yamawaki[23] finds correspondence between several psychological and computational measurements of basic texture features. They made psychological experiments to describe texture properties of typical texture patterns and then derived a set of computational features corresponding to the human visual perception: coarseness, contrast, directionality, line-likeness, regularity, and roughness. Experimental results on the texture discrimination showed a good performance.

It is useful to note here that differences in the first- or second-order statistics allow texture discrimination for a human subject, but differences in the third-order or higher-order statistics are irrelevant as long as discrimination is concerned[24]. A new method for texture discrimination will be discussed in this thesis, which is essentially a second-order statistics of gray level distribution. The measurement to be derived does not depend on a particular size and orientation, and does not need preprocessing procedures such as historgram equalization, gradient operation, and so on.

1-2 Hardware Implementation of Local Parallel Picture Processing

What kinds of picture processings are more expensive in general-purpose digital computers ? What types of computational processes are required in most picture processing operations ?

Let us consider a typical computation in picture processing techniques[25], which has a form of sum-of-products:

$$G(x,y) = \sum_i \sum_j W_{ij} F_{ij}(x,y) \qquad (1\text{-}1)$$

where $F_{ij}(x,y)$ are neighboring points around (x,y) of an input

picture, and W_{ij} are weights. Depending upon the weights, this computation can be frequently used in picture processing operations such as: convolution used in averaging, enhancement and differentitation; correlation used in object identification, and so forth.

The local parallel processing of this type can be programmed in conventional computers as shown in Fig. 1-4. Most of local processings have three different control procedures in their programs: loop control, data access, and computations.

```
      INTEGER     IN(128,128),OUT(128,128),IW(3,3)
      DATA        IW/3,5,7, ···      /
                                          Loop/Access/Computation
      DO  10  J=2,127                     Scanning Operation (Loop)   L
      DO  11  I=2,127                     Scanning Operation (Loop)   L
        ISUM=0                            Local Operation             C
        DO  20  JJ=1,3                    Local Operation             L
        JY=J+JJ-2                         Local Operation             A
        DO  21  II=1,3                    Local Operation             L
        IX=I+II-2                         Local Operation             A
        ISUM=ISUM+IN(IX,JY)*IW(II,JJ)     Local Operation             A+C
   21 CONTINUE                            Local Operation             L
   20 CONTINUE                            Local Operation             L
        OUT(I,J)=ISUM                     Local Operation             A
   11 CONTINUE                            Scanning Operation (Loop)   L
   10 CONTINUE                            Scanning Operation (Loop)   L
```

Figure 1-4 Process Flow of Sum-of-Products Operation by FORTRAN Program

With reference to Eq. (1-1), an input picture F is given by IN, an output G by OUT and a weight matrix W by a 3x3 IW.

(i) Program Loop Control

This controls two program loops: one for scanning a local operation all over a picture and the other for performing the operation in the local area. These loop controls might be automatically

done.

(ii) Data Access

This consists of data transfers between input picture and weight matrix, and output picture. The number of data accesses is exactly proportional to the size of local operation window and has a strong effect on execution time. This might be done in parallel.

(iii) Computation

This calculates the sum-of-products with weights. If weights are represented by 1 or 2's power, the sum-of-products can be implemented by adding with bit shift manipulations rather than by straight computation. It might still be required to perform computations in parallel with parallel data access in the local area.

Table 1-1 lists the distribution rate of the execution time of three different procedures in the sum-of-products operation with a 3x3 weight matrix.

	Loop	Access	Computation
1	23	30	47
2	52	35	13
3	14	53	33

(1) Exact Sum-of-Products (arbitary weights)
(2) Averaging (unit weights)
(3) Laplacian (weights : 1 and -4)

* executed by a minicomputer TOSBAC-40C

Table 1-1 Execution Time* Distribution Rate with 3x3 Weight Matrix

It has been recognized that conventional general-purpose digital computers are inefficient even for relatively simple local parallel operations. Processing of picture data by sequential computers, which are based on the so-called Von Neumann architecture, requires a large amount of computing time. The reason of the unsuitability which results in the high computing costs, is the sequential control structure. In conventional computers, programs and data are stored in the same memory unit and all operations are serially executed, but picture operations are usually highly parallel in nature.

Since two dimensional picture data arrays require large amounts of data storage and often exceed the main memory capacity. As a result, overhead time to transfer picture data between the main memory and secondary storage is large.

Therefore, it has been expected to implement cost-effective picture processing by developing a specialized hardware that is optimized for picture processing. Several picture processing machines have been proposed and some of them have been implemented. It is becoming more practical to realize such a special hardware at a modest cost with modern digital technologies.

Advanced memory technology can establish a memory storage large enough to contain digital pictures[26]. Memory chips with 16K bits or more have become commercially available. Rapid computation of the sum-of-products operation in Eq. (1-1) is of no difficulty per se. Several kinds of one-chip arithmetic units, such as adder, multiplier, multiply-sum etc., have been fabricated. Besides, digital circuit techniques can speed up computing time by parallel and/or pipeline implementation.

Here arise the following problems for the implementation of the picture processing in hardware:

(1) Picture Processing Machine Architecture

What should be the computer architecture of picture processing machines in terms of cost-effectiveness ? Trade-offs in various aspects (eg. speed, flexibility, and economy) should be discussed.

(2) Picture Memory Storage

Usually, two dimensional picture data is too large to fit into the main memory of the computer, and secondary storage must be used to contain the picture. This results in large amounts of overhead time in order to transfer picture data between the secondary storage and the main memory.

A picture processing machine should have a memory large enough to contain pictures.

(3) Important Picture Processing Functions

What computational processes are required in most picture processing operations ? What kind of computations are most frequently used ? These important functions must be first speeded up by the hardware.

(4) Local Parallel vs. Fully Parallel

There are two types of parallelism in hardware realization of picture processing: local parallel and fully parallel implementations. Processing speed, hardware complexity, and economy are to be considered.

(5) Specialized Machine vs. Flexible Control Machine

Even though several picture processing functions can be realized in hardware, it is still required to perform flexible operations by combining several basic ones. Versatile control architectures must be discussed in order to establish more powerful and flexible picture processing functions.

In this thesis, a microprogrammable local parallel picture processor is realized in a simple hardware, which can perform several basic picture processing functions at high speed. The

operations, which are frequently used in most picture processing studies, such as two dimensional convolution, data conversion, logical filtering, region labeling, affine coordinate transformation, and histogram computation, are implemented in local parallel architecture. Pixelwise arithmetic and logical operations can be programmed by microprogram instructions.

Related Works

Digital picture processing is performed on hardware ranging from general-purpose computers to special-purpose picture processing machines.

Several picture processing systems have been built on general-purpose computers (from μ-computers to very large scale computers), where picture processings are simulated by software. Main efforts have been made to develop special I/O devices for pictorial information, find efficient algorithms for picture manipulations, and widen application fields. While several papers have been published on these problems, they are not referred to here.

The objective here is to refer selectively to the works on special processors for picture processing. Fu[27] gave an excellent survey of special computer architectures for picture processing. (General surveys of parallel processors and processing can be found in other publications[28][29].)

Unger[30][31] pointed out that general-purpose computers are not good at solving problems where the data is arranged in a spatial form, and proposed a spatial computer SPAC. SPAC consists of a rectangular array of logical modules directed by a master control unit. Each module consists of a one-bit principal

register, a set of one-bit memory, and processing circuit. Each module is interconnected with its four neighbors (above, below, left, and right). The master control issues identical commands to all modules in the array. SPAC was simulated for a 36x36 array of modules, by which character patterns were processed to smooth, find edge sequence, check for concavity etc.

Unger's machine has influenced the following fully parallel array processors: ILLIAC III, Parallel Picture Processing Machine PPM, and Celluar Logic Image Processor CLIP.

McCormick[32] designed the Illinois pattern recognition computer (ILLIAC III) especially for automatic scanning and analysis of massive amounts of bubble-chamber negatives. The system consisted of input scanners, pattern articulation unit (PAU), taxicrinic unit, arithmetic unit and input complex. PAU consists of an iterative array of 1024 identical processing elements called stalactites, which are arranged in a 32x32 two dimensional array. The stalactites are interconnected in either rectangular or hexagonal array form, and are designed to perform simple Boolean and threshold operations which can be combined together to perform local parallel processings on the input picture, such as thinning, noise cleaning, line element detection, and so forth.

ILLIAC III represents a pioneering effort in realizing a special processor for picture processing. The hardware implementation was, however, complicated due to the technology available at the time.

Kruse[33] developed a parallel picture processing machine PPM which consists of a 64x64 array of identical modules, each connected to its nearest 8 neighbors, and controlled by a master control unit. Each module is a synchronous sequential circuit that takes its input from the neighboring eight modules and delivers its output to the same modules. The output and transition states of the modules are computed simultaneously over the entire array.

PPM can perform local parallel operations for gray scale pictures as well as for binary ones: logical, arithmetic, shift and transfer operations. PPM has been realized in a local parallel basis. The system logically behaves like a fully parallel machine, though the processing is sequentially done. The PICAP[34][35] (modified version of PPM) has been improved by adding a large set of measurement capabilities such as neighborhood counting, X-Y extent determination, gray scale statistics (max., min., average), and histogram computation.

Duff and his colleagues proposed and constructed a series of cellular logic image processor CLIP in order to optimize parallel array architecture and cellular logic processor design for applications in pictorial pattern recognition.

They constructed a fixed-function special-purpose processor UCPR 1[36] for use in the analysis of bubble-chamber photographs. CLIP 1[37] served to test the feasibility of constructing an integrated array processor where three picture processing functions could be implemented: extraction of connected objects, extraction of object boundaries, and extraction of the contents of closed loops. CLIP 2[38] was the first programmable array processor; A 16x12 hexagonal array receives a binary pattern on its 192 input wires. Each cell is programmable to provide two independent binary outputs which are Boolean functions of the two inputs to the cell. CLIP 3[39] was designed using the experience gained in operating these arrays. The array interconnection can be either square with eight neighbors, or hexagonal with six neighbors. A hybrid CLIP 3 array was built to gain experiences with larger picture area. 96 x 96 gray scale picture data can be handled by a hardwired scanning unit and CLIP 3 in a minicomputer system.

Using the results obtained from CLIP 3 and the hybrid version, an N-MOS LSI processor, called CLIP 4[40], with a 96x96 cell array has been built. CLIP 4 is intended for gray level processings

as well as for binary operations. A large number of programs have been written: smoothing, thresholding, contour extraction, thinning, perimeter measurement, histogramming, and so forth.

CLIP 4 is an encouraging step towards practical realization of fully parallel array picture processors. But it still has some drawbacks and limitations in the cases of local operations with larger windows, arithmetic operations on gray scale numbers etc.

These picture processing machines, as described above, are fully parallel array processors with identical processing modules arranged in two dimensional array. These machines can be implemented on a fully parallel basis using LSI technology, but they have problems on the cost and limitation of their capabilities: for example, the picture size to be processed. As Kruse mentioned, the combination of local parallel processing and sequential scanning control minimizes the cost, and fits technical characteristics of picture I/O devices and memory.

Yachida, Tomita and Tsuji[41] developed a high speed pattern processor HSPP for scene analysis. HSPP has the following features on hardware implementation: two dimensional memory (64 x 64) with 3 different access modes, 5x5 local window memory for high speed buffer memory between two dimensional memory and arithmetic/logic unit ALU, and microprogrammable controller. A 5x5 local window can be addressed either in raster scan, random scan or line track mode over two dimensional memory. Local processings include arithmetic and logical operations such as correlation, differentiation, noise cleaning, thinning, and so on.

HSPP has the flexibility to accomplish more complex picture processings in combination of basic operations, using dynamically rewritable microprogram architecture. HSPP has been used: for preprocessing such as averaging, spatial filtering etc.; for feature extraction, such as line detection, thinning, topological feature point detection etc.

While HSPP has a single processing element (ALU), Matsushima and his colleagues[42] developed an image processor IP with a 4x4 array of processing elements.

IP consists of two layers of picture memory (256x256) and buffer registers for a 4x4 local window, 16 processing elements (4x4), and microprogrammable controller. Each processing element can perform the following instructions: processor control, data transfer between registers, and arithmetic logical operations. IP has been used to accomplish picture processings in scene analysis[43]: for example, thresholding, spatial filtering, gradient computation, data conversion, geometric transformation, histogram computation, gray level statistics etc.

Robinson and Reis[10] developed a real-time edge processing unit EPU on the basis of Robinson's edge detection algorithm[7] by compass gradient masks. EPU can process picture data at the TV scanning rate. It is used for optimizing the parameters of a general scene analysis program.

EPU can perform sum-of-products operations over a 3x3 local window. This capability offers a variety of filtering operations, such as low pass and band pass filtering, Laplacian enhancement, and compass gradient edge detection operations. EPU also has special circuits to perform edge extraction.

Both fully parallel and local parallel picture processors are specially towards higher speed processing of picture data than conventional computers. Another approach towards high speed processing is the use of array processors or associative processors.

Allen and his colleagues[44] have developed a flexible array processor FP to increase the efficiency of data handling where certain calculations are performed repetitiously. FP is a special-purpose computing unit which features high computation performance and I/O capabilities as required for array processing problems: matching, correlation, spatial transformation, normaliza-

tion, and so forth.

FP serves the need for reducing the computing load on an existing central computing system as a peripheral. Several picture processing techniques have been applied on FP for the machine processing of remotely sensed data. FP is extended to support efficient signal level and symbolic level processings in the image understanding study. A new processor[45] is being designed, using the most advanced technology and automated design technique, as an add-on for general-purpose host computers, and includes a number of functional units.

Bouknight and his colleagues[46] have realized a large array processor ILLIAC IV to solve complex scientific problems involving arrays or matrices. ILLIAC IV consists of 64 processing elements and one master control unit. The control unit executes instructions in parallel with processor elements which can perform a wide range of arithmetic and logical operations. ILLIAC IV is under control of a general-purpose machine which handles all I/O and system monitoring tasks in a conventional manner.

ILLIAC IV is a powerful array processor for arithmetic operations with floating point data on matrices, vectors etc. Therefore, the system is most suited for such calculation intensive applications as two dimensional FFT, image restoration, and warp processing.

Batcher and his colleagues[47] designed a large scale associative processor STARAN, which consists of a control system and a number of associative array modules. Each array module contains a 256-word x 256-bit multi-dimensional access memory, 256 simple processing elements, a permutation network and a selector. The array module and permutation network permit memory access in different modes, for example, data access in either column mode or row mode of two dimensional signal. Hence, it is possible to process either all bits within a word in parallel or the same

field of all desired words in parallel. Shifting capability is also used for various data manipulations such as fast Fourier transform and warp processing[48].

We have seen that there are several types of picture processing machines: fully parallel picture processors, local parallel picture processors, array processors, and associative processors. Besides these processors for general picture processing, some specially designed hardware have been developed for the analysis of remotely sensed multispectral data[49][50], microscopic cell data[51], character patterns[52], and so on.

From the points of view of cost, speed and capability, a local parallel picture processor was developed, where some of most frequently used picture operations were implemented in a simple hardware. Other picture processings can be accomplished by a microprogrammable architecture.

This thesis will enhance software implementation techniques of local parallel picture operations for edge detection and texture analysis, as well as hardware realization of a versatile local parallel picture processor.

In Chapter 2, local parallel picture processings are realized for edge detection and texture analysis problems of current concern. Section 2-2 discusses edge detection procedures by differentiation operations in terms of local area size and shape as well as computational complexity. Two independent studies were done with pictures of human face. Differentiation of Laplacian combined with averaging yields good edge information. The second study is concerned with a nonlinear operation (maximum detection) by a set of forward, backward and central derivatives. A new thresholding

technique was employed therein to improve line-likeness in the resultant edge picture.

These two edge detection procedures provide a simple low level picture processing algorithm to obtain a good sketch from a given input picture. They are independent of any a priori knowledge in the picture, and they lead to pattern recognition problems.

A new texture analysis method by local operation is introduced in Section 2-3. Texture parameters are calculated to identify a picture by simulating planar random walks in a local area. The planar random walk can be implemented either in a mathematical way or in a simulation with the random number generation. This localized texture analysis procedure gives a set of statistical information which indicates gray level distribution in the two dimensional sense.

These two typical picture processing algorithms on edge detection and texture analysis were realized in computer programs and tested on a number of picture data. Software implementation techniques give an aspect of speeding up local parallel operations in terms of program loop control, picture data access control, and computation in a local area.

In Chapter 3, a microprogrammable local parallel picture processor is implemented in a simple hardware, which can perform several basic functions at high speed. Section 3-1 discusses major design concepts for realizing the picture processing machine. The picture processor is directly connected to the picture memory to minimize data transfer between memory hierachies. The operations which are most frequently used, such as convolution, data conversion, logical filtering, affine coordinate transform, histogramming, and region labeling, are implemented in a local parallel architecture.

The hardware architecture is described in Section 3-2. Interfaces to a host computer and to the picture memory, microprogrammable controller, address controller and special picture

processing modules are introduced. Detailed descriptions on address controller, pixelwise operation modules, and local parallel processing modules are listed in Sections 3-3, 3-4, and 3-5, respectively.

Chapter 4 mainly discusses the software system for the picture processor, and shows some examples of programming the processor with illustrative results. From the points of view in software architecture of the picture processor, the software systems in a host computer and in the picture processor are discussed in Sections 4-1 and 4-2, respectively.

Section 4-3 explains how to program the picture processor. A repertoire of applicable picture processings are listed, and a typical program sequence is shown for calculating the gradient vector. Some experimental results in Sections 4-4 and 4-5 indicate that the picture processor is powerful enough to allow quick analysis of pictures and to explore novel applications.

CHAPTER 2 LOCAL PARALLEL PICTURE PROCESSING TECHNIQUES

In this chapter, the local parallel picture processing is first summarized. Some problems in computer implementations are then discussed. In Section 2-2, two different approaches are taken for edge detection by derivative operations. First, the Laplacian operator combined with averaging will be presented. Second, another new edge detection technique uses a set of finite differences and dynamic local thresholding. Problems on the size and shape of the local area, computation complexity and threshold determination will be mainly examined. In Section 2-3, a new texture analysis method based on the planar random walk in the local area will be studied. The local processing is extended to analyze texture patterns in a statistical way.

This chapter preludes the hardware implementation of picture processings which will be described in Chapters 3 and 4.

2-1 Introduction

The local parallel picture operation is a function whose values defined at a point depend only on a small set of its neighbors in a given picture. The value may be assigned to the corresponding point in the output picture or a value of extracted feature set.

A number of local parallel processing techniques have been studied to transform an input picture into another or to extract a

set of features from a picture.

Reasons why the local parallel processing has been given considerable attention in the computer picture processing are as follows:

(i) One of the most practicable picture processing techniques is, what is called, the local parallel processing.

(ii) It can be considered to be a simulation of the human visual system. The computer processes pictures as the human eye does in the visual system.

(iii) It is easy to implement the local parallel processing in the computer. There is no need for special picture manipulation algorithms, like the fast Fourier transform, the K-L transform etc.

Fig. 2-1 shows a typical framework of the local parallel operations which convert an input picture into another picture (output). A value at (I, J) in the output picture is obtained by a certain operation in the local area or neighborhood of the corresponding input picture element (I, J). The local area is called a "window". There are several ways in the order where the window is placed in the input picture. One of the most common ways is a raster scan, like in the TV, where the window is swept all over the picture column-by-column and then line-by-line. (See the flow of the local operation by the raster scan in Fig. 1-4.) Another way is a

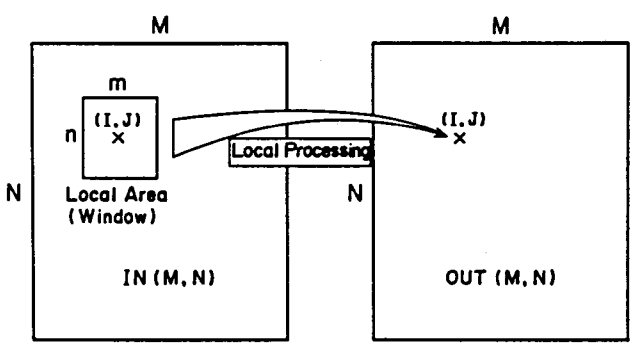


Figure 2-1 Typical Framework of Local Parallel Operation

tracing technique in which the window is moved to any direction in going from the present point to the next. The tracing method may be used to achieve efficient processing only for interesting parts. The control structure, however, will be more complex because of the multiple choices of tracing at a point and back tracking.

Several local parallel operations in the window have been developed not only for binary pictures, but also for gray scale pictures.

(1) Local Operations for Binary Pictures

Smoothing is important if one considers real data containing noise in the form of the absence or presence of the signal. One example of smoothing is to eliminate isolated islands or notches, and to fill isolated gaps by means of Boolean functions in a 3 x 3 window.

The thinned line will usually describe the essential topological aspect of the binary picture, particularly in the case of character patterns or line drawings. A thinning operation removes a black element only when the operation does not result in shortening of the length of connected black elements, or creation of a gap in it. Mostly, a 3x3 window operation is used and iterated to obtain a skeleton line.

The contour line is also used in the region description of the binary picture. The binary edge detection is simply done by leaving "1" elements which touch "0"s and by eliminating "1"s which are completely surrounded by "0"s.

Topological features are important attributes of objects in the pictures. In character recognition, a 3 x 3 window is often used to find the direction of line segments or to label the characteristic points like crossing, branching and ending points.

(2) Local Operations for Gray Scale Pictures

A low-pass filter or local averaging operation is used to

eliminate random speckled noise and abrupt changes in brightness. The averaged picture, analogous to the one of out-of-focus, can be coded and transmitted with fewer samples than the original picture. This local operation employs the convolution or sum-of-products computation.

Edge or contour detection is one of important techniques to extract from a picture information which is significant to recognize or classify objects. Local differential operators are used to produce edge or contour lines, which can be implemented by the sum-of-products computation. The difference between averaging and differentiating is in the weighting coefficients; especially the algebraic signs are different. Various digital weighting masks have been proposed to realize the gradient, Laplacian and other derivative operators.

Living visual systems are known to subjectively enhance sharp differences in gray level; i.e., Mach phenomenon[53].

The differential operator can be used to simulate this edge enhancement phenomenon.

The detection and identification of objects in a given picture are frequently accomplished by the correlation technique. The correlation operation calculates the similarity between a picture and a standard or representative pattern. This type of approach is often termed "template matching", or "matched filtering".

Other local operations for the object enhancement and extraction are found in Prewitt[54] and for the feature extraction in Levine[5].

From the technical point of software implementation, the efficiency of local parallel processing depends on the size, shape and orientation of the window, the computational form, and the control structure. Most primitive edge detectors use square windows, whose size falls between 3x3 and 16x16. The Hueckel's operator

is defined on a circular disc with diameter values ranging from 6 to 16. Most typical computational forms are Boolean functions for binary pictures and the sum-of-products for gray scale pictures. Other operations are based on nonlinear mathematical computation, statistical estimation,and logical decision. The control structure is divided into the local operational control and the scanning control. The local operation includes data access and computation, while the scanning control is how to sweep it all over the picture.

Following two sections will demonstrate how to implement the local parallel processing for edge detection and texture analysis. These two studies will be a preparatory step for hardware implementation.

2-2 Edge Detection Using Local Parallel Processing

An edge element represents the boundary between two adjacent regions within a set of neighboring picture elements in terms of at least one feature, for example, gray level. The decision on whether or not a picture element is on an edge is made on the basis of the gray levels in the local area.

A method to detect edges is based on the differentiation, or derivative operation in the local area which responds to sharp changes in the gray level. Two different approaches, based on simple derivative operators, will be discussed in this section mainly from the point of software implementation. An edge detection procedure consists of the derivative operation and threshold selection. Two edge detection techniques are studied in terms of derivative types, local area size and shape, and thresholds.

The first edge detector is derived from the second derivative

(the Laplacian operator) and thresholding by a predetermined value. Effects of local area size and threshold values are considered. The second edge detector consists of a set of derivative operators (first, second and third),and dynamic thresholding technique. Some experiments demonstrate how the selection of derivative operations and threshold affects the results.

2-2-1 Laplacian Edge Detector

Let us consider the Laplacian operator. The Laplacian ∇^2 is defined as

$$\nabla^2 f(x, y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \qquad (2\text{-}1)$$

where f(x,y) represents a gray scale picture or two dimensional signal of the two spatial coordinates x and y. The Laplacian is the second derivative, while the gradient operator is the first one. The gradient ∇ is defined as

$$\nabla f(x,y) = \frac{\partial f(x,y)}{\partial x} u_x + \frac{\partial f(x,y)}{\partial y} u_y \qquad (2\text{-}2)$$

where u_x and u_y are unit vectors in the x and y directions, respectively.

As seen in Eqs. (2-1) and (2-2), the $\nabla^2 f$ signal is a scalar and thus easier to store and treat in a computer than the ∇f, which is a vector signal and has two components.

The Laplacian is a more efficient representation of the edge signal than the gradient, in the case where the edge is extended like a ramp. As Fig. 2-2 indicates, the Laplacian takes large values only at the points where the slope changes, while the gradient signals all along the slope. (For a convenience sake, Fig.2-2

shows a ramp-like edge in one dimension.) In addition, the Mach effect in the visual system is known to be present at the points of inflection rather than along the slope as shown in Fig. 2-3. This phenomenon results in the subjective enhancement of sharp brightness changes, which could be approximated by subtracting the second derivative from the signal.

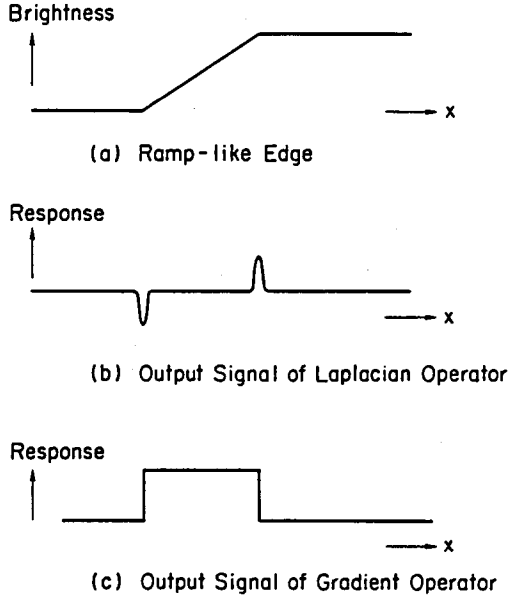


Figure 2-2 Responses to Ramp-Like Edge by Laplacian and Gradient Operators

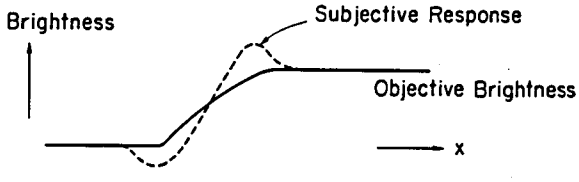


Figure 2-3 Mach Phenomenon

The Laplacian, however, may be less immune to noise in the mathematical sense since it involves a higher derivative.

In digital processing, the Laplacian can be replaced by the finite difference as follows:

$$\nabla^2 f(i,j) = f(i+1,j) + f(i-1,j) + f(i,j+1) + f(i,j-1) - 4f(i,j) \quad (2\text{-}3)$$

where $f(i,j)$ is a sampled picture taken from $f(x,y)$.

Eq. (2-3) can be derived in the following way. The first directional (partial) derivative $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$, as in the gradient of Eq. (2-2) can be written by

$$D_x f(i,j) = f(i,j) - f(i-1,j) \quad (2\text{-}4)$$

and

$$D_y f(i,j) = f(i,j) - f(i,j-1). \quad (2\text{-}5)$$

Similarly, the second derivative $\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ becomes

$$\begin{aligned}
\nabla^2 f(i,j) &= D_{x2} f(i,j) + D_{y2} f(i,j) \\
&= D_x \{D_x f(i,j)\} + D_y \{D_y f(i,j)\} \\
&= \{f(i+1,j) - f(i,j)\} - \{f(i,j) - f(i-1,j)\} \\
&\quad + \{f(i,j+1) - f(i,j)\} - \{f(i,j) - f(i,j-1)\} \\
&= f(i+1,j) + f(i-1,j) + f(i,j+1) + f(i,j-1) - 4f(i,j).
\end{aligned} \quad (2\text{-}6)$$

Fig. 2-4 depicts the Laplacian operator in the schematic form. This is a primitive mask to perform the second derivative operation in a digital picture. The Laplacian signal $\nabla^2 f$ is produced by convolving the operator mask with pictures, that is, by the sum-of-products computation.

The detection of true edges in a picture is not easy. It depends on how large is the change in brightness, how fine is the

details, and how much random noise are contained. The edge detection procedure includes some sort of nonlinear operations, such as thresholding, to determine if an edge point is important. Fine details of the picture can be sensed by smaller windows, while coarse structured edges can be detected by larger windows. Any efforts to reduce the noise content of the original picture will have a remarkable effect on the edge detection.

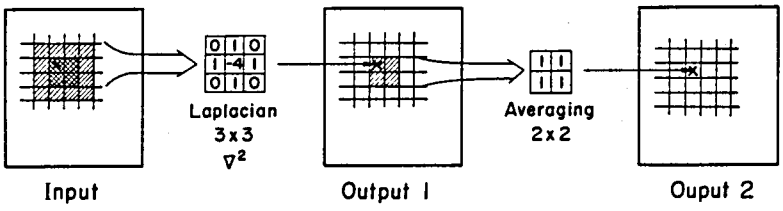
0	1	0
1	-4	1
0	1	0

Figure 2-4 Laplacian Operator

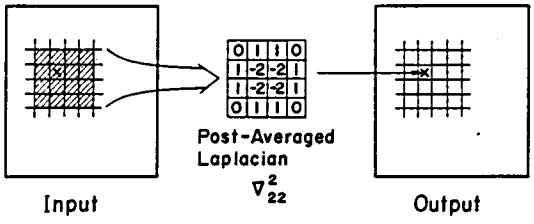
Two different approaches have been developed to reduce noise by the averaging operation. The first one, a post-averaged Laplacian, cascades the Laplacian operation and the averaging operation into one operator. A 3 x 3 local window is used for the Laplacian and smaller window for averaging, for example, 2 x 2, 3 x 3 etc. The other, a running Laplacian, combines the Laplacian with averaging at each Laplacian cell. Each cell of a running Laplacians has a certain size of picture elements, for example, 2 x 2, 3 x 3 etc.

Post-Averaged Laplacian

Fig. 2-5 depicts the conceptual model of a post-averaged Laplacian, where the Laplacian is first performed on an input picture and then its resultant picture is averaged by a local mask. The Laplacian is a 3 x 3 basic operator as shown in Fig. 2-4, and the averaging operator consists of a 2 x 2 local mask with unit weights. These consecutive operators can be composed into a larger Laplacian as follows.



(a) Serial Operation



(b) Composite Operation

Figure 2-5 Composition of Post-Averaged Laplacian ∇^2_{22}

Let $\nabla^2{}_{mn}$ denote a larger Laplacian, where double subscript (m, n) indicates the size of local window for post-averaging. A post-averaged Laplacian ∇^2_{22}, as shown in Fig. 2-5, is given by composing two operations:

$$\nabla^2_{22} f(i,j) = \sum_{k=i}^{i+1} \{f(k,j-1) + f(k,j+2)\} + \sum_{\ell=j}^{j+1} \{f(i-1,\ell) + f(i+2,\ell)\} - 2 \sum_{k=i}^{i+1} \sum_{\ell=j}^{j+1} f(k,\ell) \,. \tag{2-7}$$

A ∇^2_{22} can be interpreted as operating the Laplacian ∇^2 over a 2 x 2 local window, simultaneously. (The basic Laplacian ∇^2 can be considered as a ∇^2_{11} which has only a single cell for averaging, i.e., no averaging.)

Eq. (2-7) can be written by,

$$\nabla^2_{22} f(i,j) = \nabla^2 f(i,j) + \nabla^2 f(i+1,j) + \nabla^2 f(i,j+1) + \nabla^2 f(i+1,j+1) \,. \tag{2-8}$$

Fig. 2-6 shows the illustrative design concepts of post-averaged Laplacians ∇^2_{11} and ∇^2_{22}. Similarly, post-averaged Laplacian with larger sizes of local window (3x3, 4x4, and so on) can be obtained. If necessary, non-square operators may be defined such as ∇^2_{12}, ∇^2_{21}, ∇^2_{23} etc., whose sizes differ in the horizontal and vertical directions. Fig. 2-7 illustrates ∇^2_{33} and ∇^2_{12}.

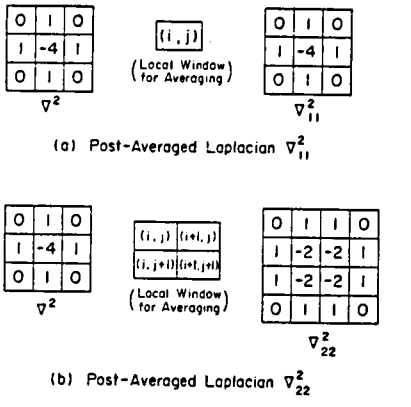


Figure 2-6 Post-Averaged Laplacians ∇^2_{11} and ∇^2_{22}

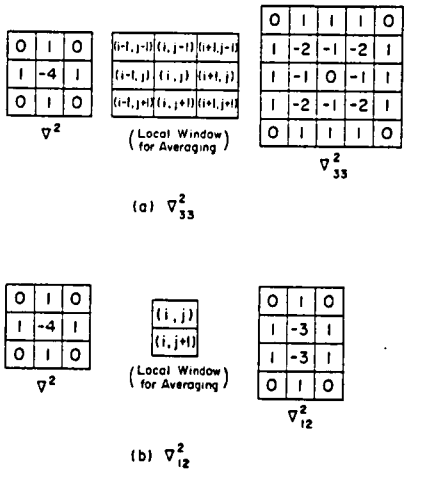


Figure 2-7 Post-Averaged Laplacians ∇^2_{33} and ∇^2_{12}

Running Laplacian

Averaging is simultaneously performed on each cell of the Laplacian. In other words, a "running Laplacian" has a certain size of local window, corresponding to each Laplacian operator cell, over which the averaging is done. Fig. 2-8 illustrates a running Laplacian ∇^2_{wn}; the double subscript (w, n) indicates the size of Laplacian cell in picture elements.

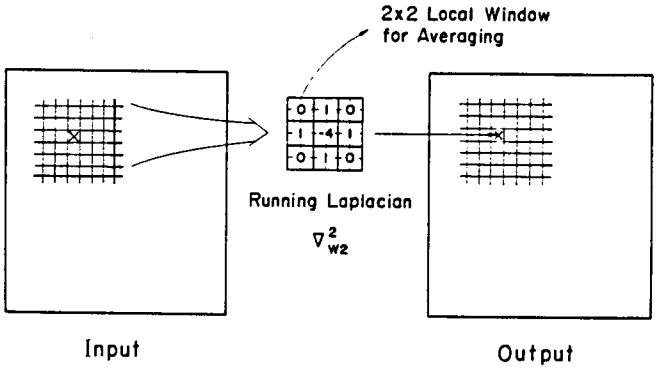
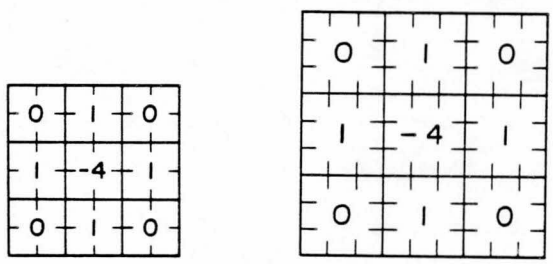


Figure 2-8 Composition of Running Laplacian ∇^2_{w2}

Therefore, ∇^2_{w2} is given by

$$\nabla^2_{w2} f(i,j) = \Big(\sum_{k=i}^{i+1} \sum_{\ell=j-2}^{j-1} + \sum_{k=i-2}^{i-1} \sum_{\ell=j}^{j+1} + \sum_{k=i+2}^{i+3} \sum_{\ell=j}^{j+1} + \sum_{k=i}^{i+1} \sum_{\ell=j+2}^{j+3} \Big) f(k,\ell) - 4 \sum_{k=i}^{i+1} \sum_{\ell=j}^{j+1} f(k,\ell) \,. \qquad (2\text{-}9)$$

The basic Laplacian can be considered as ∇^2_{w1}, where each Laplacian cell has only one picture element. ∇^2_{w2} has a 2 x 2 local window for averaging at each operator cell. A further running Laplacian is ∇^2_{w3} in which 3x3 picture elements are alloted to each cell as shown in Fig. 2-9.



(a) ∇^2_{w2} (b) ∇^2_{w3}

Figure 2-9 Running Laplacians ∇^2_{w2} and ∇^2_{w3}

These Laplacian operators were applied to pictures of human face to extract the contour line. The Laplacian described above was implemented with thresholding technique which produced a binary contour line picture.

Figure 2-10 Digital Picture of a Human Face

The input picture is scanned into a 128 x 128 digital array by

means of a TV camera for photographs. Fig. 2-10 shows an example of a digital picture on the display unit.

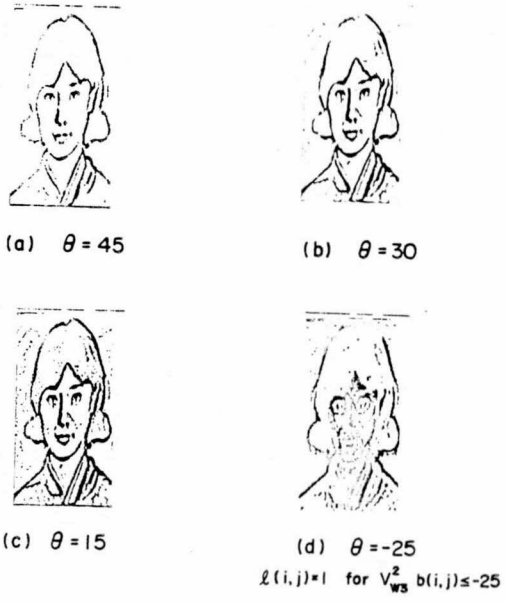
The Laplacian operator illustrated in Fig. 2-9(b), for example, is applied to the picture of Fig. 2-10. Then, thresholding the resultant picture at a given threshold value produces the binary pictures shown in Fig. 2-11.

Let l(i,j) be a binary picture obtained by the procedure mentioned above:

$$\begin{cases} \ell(i,j) = 1 \quad ; \quad \nabla^2_{w3}\, b(i,j) \geq \theta \\ \ell(i,j) = 0 \quad ; \quad \text{otherwise} \end{cases} \tag{2-10}$$

where θ is a given constant value.

Figs. 2-11 (a) through (d) depictes the binary output for the same input picture, if the threshold values θ are 45, 30, 15, and -25, respectively. The lower θ is, the more obviously a contour line grows, while more noise is superimposed. If θ is negative



(a) $\theta = 45$ (b) $\theta = 30$

(c) $\theta = 15$ (d) $\theta = -25$

$\ell(i,j)=1$ for $\nabla^2_{w3}\, b(i,j) \leq -25$

Figure 2-11 Binary Output Pictures by Laplacian ∇^2_{w3}

as in Fig. 2-11(d), the contour lines appear in parallel to those thresholded by a positive value.

Fig. 2-12 shows the output pictures with other Laplacian operators, ∇^2_{w2}, ∇^2_{33} and ∇^2_{22}, for the same input picture.

(a) ∇^2_{w2} ($\theta = 35$)

(b) ∇^2_{33} ($\theta = 15$)

(c) ∇^2_{22} ($\theta = 10$)

Figure 2-12 Binary Output Pictures by Laplacians

As compared with these binary output pictures, a running Laplacian operator ∇^2_{w3} which has a larger averaging window 3 x 3 provides a better perceptible result. It is assumed here that the size of input pictures is such that a front view face is just fixed inside the picture boundaries, and there is no shading problem in the lighting. Figs. 2-13 and 14 demonstrate the better performance by the running Laplacian operator ∇^2_{w3}.

This contour line extraction algorithm by the Laplacian was implemented in a machine language for a minicomputer TOSBAC-40C. The program was written in a straight forward manner with left shifts instead of multiplication on each pixel. The execution time for the Laplacian together with thresholding is approximately 3 seconds for a 128 x 128 digital picture. If the program is coded more carefully, the execution time should be improved.

The contour line processing of human faces can be widely

(a) Input

(b) Output

Figure 2-13 Contour Line Extraction by ∇^2_{w3}

(a) Input

(b) Output

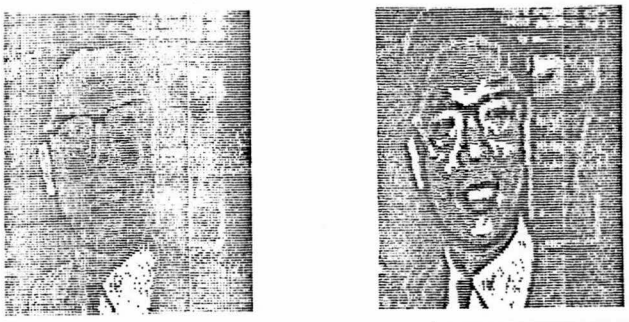
Figure 2-14 Contour Line Extraction by ∇^2_{w3}

used in application fields such as computed portrait, computerized animation, human identification[55], physiognomy, data compression etc.

Computer simulation of Mach phenomenon can be given by the use of the Laplacian operator. Enhanced picture $f^*(x,y)$ is computed by

$$f^*(x,y) = f(x,y) - m\nabla^2 f(x,y) \qquad (2\text{-}11)$$

where m is a constant. Fig. 2-15 illustrates an example by Laplacian ∇^2_{w3}.



(a) Input (b) Output

Figure 2-15 Mach Phenomenon Simulation by Laplacian ∇^2_{W3}

2-2-2 Nonlinear Edge Detector

Another edge detection approach is a nonlinear local operation. The nonlinear edge detection procedure, which is described in this section, consists of edge element detection, global thresholding, and local thresholding.

In the first step of detecting edge elements, size and shape of the local window were determined as follows. An edge operator operates on local windows of different sizes and shapes in order to detect the most prominent edge. Finite differences in higher-orders are used as edge operators. The direction type of finite differences will determine the area to be computed, while the differences order will decide the shape of local window and how the pixels within the window are to be weighted, or computation complexity.

Suppose an edge element is identified at the pixel P(x,y), and let it be the origin of a cartesian system, as shown in Fig.2-16. The local window is placed in each one of the four cartesian quadrants with the forward and backward finite differences. The central

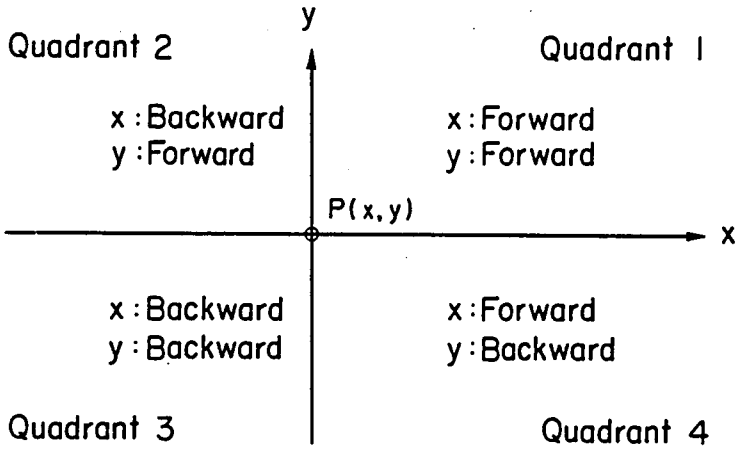


Figure 2-16 Direction Type of Difference Operations

finite difference is operated on the local window whose center point is at P(x,y).

The first-order forward, backward and central differences are defined by Hamming[56] as follows:

Forward Difference,

$$\Delta f(\vec{z}) = f(\vec{z}+\vec{h}) - f(\vec{z}) \quad , \tag{2-12}$$

Backward Difference,

$$\nabla f(\vec{z}) = f(\vec{z}-\vec{h}) - f(\vec{z}) \quad , \tag{2-13}$$

Central Difference,

$$CD f(\vec{z}) = f(\vec{z}+\vec{h}) - f(\vec{z}-\vec{h}) \quad , \tag{2-14}$$

where f is the picture function, $\bar{z} = (x,y)$ stands for pixel location and $\bar{h} = (h_x, h_y)$ is the interval size. Fig. 2-17 shows these three difference operators. The central difference will provide the continuity between forward and backward operations to detect edge elements.

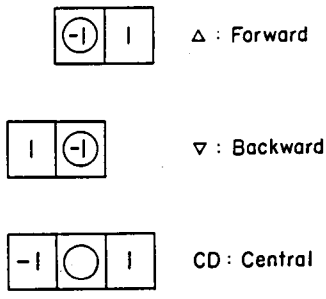


Figure 2-17 First-Order Forward, Backward and Central Difference Operators

○ indicates the pixel position, P(x,y) of interest

The above formulas can be generalized as the n-th order forward, backward and central difference, respectively. That is,

$$\Delta^n f(\bar{z}) = \sum_{k=0}^{n} (-1)^k \binom{n}{k} f\{\bar{z} + (n-k)\bar{h}\} \quad , \tag{2-15}$$

$$\nabla^n f(\bar{z}) = \sum_{k=0}^{n} (-1)^k \binom{n}{k} f\{\bar{z} - (n-k)\bar{h}\} \quad , \tag{2-16}$$

and

$$CD^n f(\bar{z}) = \sum_{k=0}^{n} (-1)^k \binom{n}{k} f[\bar{z} + g\{(n-2k)/2\}\bar{h}] \quad , \tag{2-17}$$

where

$$g(\alpha) = \begin{cases} \lceil \alpha \rceil & ; \quad \text{if} \quad \alpha \geq 0 \\ \lfloor \alpha \rfloor & ; \quad \text{otherwise} \end{cases}$$

$\lceil \alpha \rceil$ and $\lfloor \alpha \rfloor$ are the ceiling and floor operators, which are defined as the first integer value above and below their arguments, respectively. For example, the second-order (n = 2) differences are given by

$$\Delta^2 f(\vec{z}) = \sum_{k=0}^{2} (-1)^k \binom{2}{k} f\{\vec{z} + (2-k)\vec{h}\}$$

$$= f(\vec{z} + 2\vec{h}) - 2f(\vec{z}+\vec{h}) + f(\vec{z}) \quad , \qquad (2\text{-}18)$$

$$\nabla^2 f(\vec{z}) = f(\vec{z} - 2\vec{h}) - 2f(\vec{z} - \vec{h}) + f(\vec{z}) \quad , \qquad (2\text{-}19)$$

$$CD^2 f(\vec{z}) = f(\vec{z}+\vec{h}) - 2f(\vec{z}) + f(\vec{z} - \vec{h}) \quad , \qquad (2\text{-}20)$$

and shown in Fig. 2-18.

The forward and/or backward finite difference operator is applied in x and y directions of the quadrants as shown in Fig. 2-16. The direction type of finite difference can be defined as {q /1, 2, 3, 4, 5} if it is evaluated in quadrant 1, 2, 3, and 4, respectively, or if evaluated in 5 for the central difference.

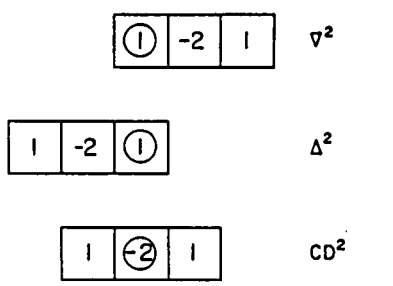
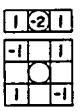
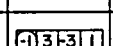
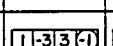
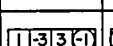
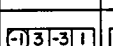
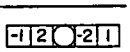
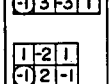
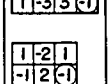
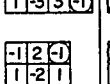
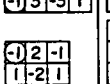
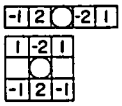
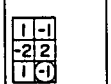
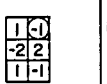
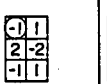
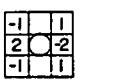
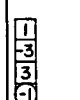


Figure 2-18 Second-Order Difference Operators

○ indicates the pixel position of interest.

The order of finite differences is limited up to three because of noise characteristics and computational convenience. As the order of the difference increase, so does the size of the local window with the shape change. The order of difference can be defined as {d /1, 2, 3, 4, 5, 6, 7, 8, 9} if these differences are enumerated as D_x, D_y, D_x2, D_{xy}, D_y2, D_x3, D_x2_y, D_{xy}, and D_y3. Table 2-1 lists all the finite difference masks in the direction and order.

Once all the finite differences are evaluated, a best edge element which yields a maximum difference value v at the pixel P(x,y) can be characterized by a pair of direction and order types of finite difference (q, d). The output m(x,y) is given by the tuple (q, d, v) which contains information about the direction and order of differ-

Order of Finite Difference (d)		Quadrant (q) of Finite Difference 1	2	3	4	5
		Δxf , Δyf	∇xf , Δyf	∇xf , ∇yf	Δxf , ∇yf	CDf
1	D_x					
2	D_y					
3	D_{x^2}					
4	D_{xy}					
5	D_{y^2}					
6	D_{x^3}					
7	D_{x^2y}					
8	D_{xy^2}					
9	D_{y^3}					

○ indicates the pixel position of interest.

Table 2-1 Finite Difference Operators

ence, and its value to indicate the edge likeness,

$$m(x, y) = (q, d, v) . \tag{2-21}$$

The second step is the global thresholding. It aimed at eliminating spurious edge elements which are mainly due to small variations in the gray level value. Global threshold t can be determined in the following way. A histogram function H of v can be obtained all over the output picture. Then t is chosen as the first local maximum given by

$$t(v) = \frac{|H(v) - H(v-1)|}{\min\{H(v), H(v-1)\}} . \tag{2-22}$$

The purpose is to identify the peak of the histogram corre-

sponding to the spurious edge elements, which have smaller difference values in the homogeneous region. Once t is determined, output m(x,y) is thresholded at each pixel, as follows:

$$m(x, y) = \begin{cases} (q, d, v) & ; \quad \text{if } v \geq t \\ (q, d, 0) & ; \quad \text{otherwise .} \end{cases} \tag{2-23}$$

The global thresholding operation is equivalent to a binary decision by which meaningless edge elements are eliminated for any further consideration.

The third step is the local thresholding. The edges usually involve gradual transitions of gray levels rather than sharp ones. The thresholding operation by a constant value usually produces thicker edges than desired. The proposed local thresholding will cope with this additional thinning problem and will be also insensitive to the shading problem of a given picture.

The local thresholding operation, combined with thinning, is performed over a number of local windows intersecting at a given pixel P(x, y). The local windows A(i) are all the 5x5 arrays that include pixel P(x, y), as shown in Fig. 2-19. There are 25 such windows. Histogram H_i is built on the difference value v at each local window A(i). If value v(x,y) at pixel P(x,y) under consideration is ranked among the highest third of the values of the histogram H_i, the pixel is considered as having "passed" the threshold test, and identified as a candidate of edge element. This thresholding operation is performed for all the histogram ($H_1 \sim H_{25}$). If pixel P(x,y) fails

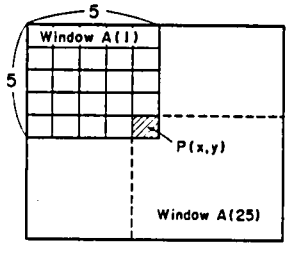


Figure 2-19 A Set of Windows for Local Thresholding

any of the above thresholding tests, the local thresholding with thinning is finished and the output configuration is set to $(q, d, 0)$. If pixel P(x,y) succeeds through the sequence of all 25 local windows, its output configuration is set to $(q, d, 1)$ as an edge element.

The finite differences can be implemented by using the mask shown in Table 2-1. It should be pointed out that the difference masks are differently weighted in order to facilitate comparisons between the outputs for finite differences of different orders. The direct comparison of the value of finite difference operators is meaningless unless we take into account the number of pixels used in computing them. For example, as can be seen from Table 2-1, D_x and D_y use two picture element values, D_{x2}, D_{xy} and D_{y2} use four pixel values (in the case of D_{x2} and D_{y2} one of them is used twice), and D_{x3}, D_{x2y}, D_{xy2} and D_{y3} use eight pixel values, if the forward and backward differences are evaluated, and six pixel values, if the central difference is the one to be evaluated for D_{x3} and D_{y3}. Therefore, in order to equally consider the outputs of different operators, the masks must be weighted according to the number of pixels involved in the computation.

Regarding Table 2-1, the pixel at which the finite difference is calculated is inscribed in a circle. Weights W are defined as follows:

$$W(q,d) = \begin{cases} 1 & ; \text{ for } d = 1, 2 \\ \frac{1}{2} & ; \text{ for } d = 3, 4, 5 \\ \frac{1}{3} & ; \text{ for } d = 6, 9 \text{ and } q = 5 \\ \frac{1}{4} & ; \text{ otherwise .} \end{cases} \qquad (2\text{-}24)$$

Once the weighted masks have been convolved with the picture data for each pixel, it is easily found which difference operator achieves the maximum value according to Eqs. (2-15), (2-16) and (2-17).

The sequence of pictures corresponding to the edge detection steps shows the metamorphosis that a picture undergoes. The metamorphoses the input picture would have gone through if some parameters in the edge detection algorithm had been modified are also demonstrated.

Fig. 2-20(a) shows an input picture of 128x128 pixels. The output for this picture after three steps (i.e., differentiation, global thresholding, and local thresholding), is depicted in Fig. 2-20(b). For comparison, Fig. 2-20(c) shows the output for the same input picture if the second step is omitted (no global thresholding). A comparison between Figs. 2-20(b) and (c) indicates the advantage of global thresholding as a means of eliminating some noise.

(a) Input (b) Output with Three Steps (c) Output without Global Thresholding

Figure 2-20 Comparative Study of Global Thresholding in Nonlinear Edge Detection

Several modifications of the parameters for the third local thresholding step were done, and the results are shown in Fig. 2-21. Figs. 2-21(a), (b) and (c) depict the outputs after the third step for the same input if the size of the window and the percentage of top-ranked pixels for the local thresholding are 5x5 and 25%, 7 x 7 and 25%, and 7x7 and 33%, respectively. The results for a 3x3 window were very poor and are not shown here.

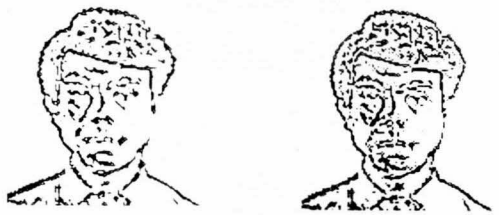
(a) 5x5 and 25% (b) 7x7 and 25% (c) 7x7 and 33%

Figure 2-21 Local Thresholding Techniques

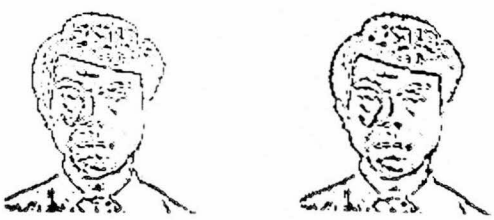
Another variation in the local thresholding operation is use of one window with pixel P(x, y) as a center. This simplified operation was tested for configurations of the following window sizes and percentages: 5x5 and 25%; 5x5 and 33%; 7x7 and 25%; and 7x7 and 33%. The results are shown in Figs. 2-22(a) - (d), which indicate that the simplified local thresholding is too local in its decision, as compared with the described algorithm.

Figs. 2-23 and 2-24 display the input digitized pictures and results.

The procedure described above was implemented on a PDP-11/45 minicomputer system. The implementation was written in a machine language program where the computation of finite differ-



(a) 5x5 and 25% (b) 5x5 and 33%

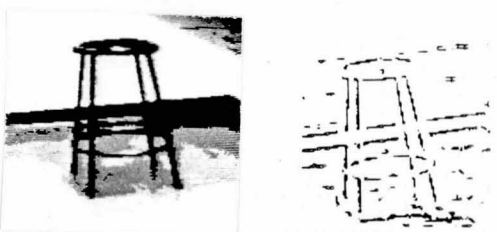


(c) 7x7 and 25% (d) 7x7 and 33%

Figure 2-22 Comparative Study of Simple Local Thresholding

(a) Input (b) Output

Figure 2-23 Nonlinear Edge Detection (Face)



(a) Input (b) Output

Figure 2-24 Nonlinear Edge Detection (Stool)

ence was executed mostly by right and left shifts instead of division and multiplication. The total execution time was approximately half a minute.

A comparison study between the Laplacian operator and non-linear operation was done using the photograph of a face shown in Fig. 2-20(a). Fig. 2-25 gives two outputs by Laplacian ∇^2_{w3} and non-linear edge detection procedure.

The original picture of Fig. 2-20(a) has a slight brightness variation between the left and right face contour. In this case, the non-linear edge detector has demonstrated by usefulness of a

(a) Output by Laplacian ∇^2_{W3} (b) Output by Non-linear Operator

Figure 2-25 Comparison of Laplacian and Non-linear Edge Detectors

flexible threshold determination in each local area.

2-3 Texture Analysis Using Local Processing

Texture can be defined simply as several unit patterns (primitives) scrambled over a picture according to certain placement rules. Texture analysis is usually performed using either a statistical or structural approach. The statistical approach derives a set of local measurements for a given picture in the space domain or in the corresponding frequency domain. Features based on these measurements are then used for discrimination between different textures. The structual approach assumes that a set of primitives can be easily identified. It then defines the texture as spatial placements of such primitives. The two methods can be combined in a hierachical way, where the statistical procedure provides the raw data for the structural procedure. A new method, which is discussed here, measures statistical parameters using the local processing and then discriminates texture patterns.

The new procedure is based on planar random walks[56].

It is assumed that the transition between different regions of a picture should be represented by regions of high non-homogeneity. The possibility of a region being non-homogeneous is related to the amount of variation (change) occurring in that region. The planar random walk approach can measure the amount of variation in the local region of a given picture and subsequently assign a degree of non-homogeneity based on the distribution of pixel values over an entire picture. This degree of non-homogeneity is a clue to texture discrimination.

(1) Planar Random Walks

A random walk in the plane is characterized by a particle moving in the unit step toward one of the four directions (left, right, up, and down) if 4-neighbor connectivity is assumed. (The procedure could be easily extended to the 8-neighbor connectivity.) The random walk is fully specified,if the probabilities of particle movements are defined, by which a particle leaves a given pixel and moves to any one of the neighboring pixels.

Let A be a local window of size 5x5 within a given picture. The size of window should be chosen so that the window is big enough for measuring texture parameters and small enough not for interfering texture discrimination. If the random walk is performed over window A, then the absorbing barrier is defined as shown in Fig. 2-26. This means that once a particle hits the barrier it is absorbed there and its random walk stops. It is easy to show that each particle is eventually absorbed by the barrier with probability 1 in a random walk. In Fig. 2-26, point B_j indicates a boundary point on the barrier, while point M_i is a mesh point inside of the barrier.

Now let us consider the moving probabilities related to the planar random walks. It is obvious that the probabilities of leaving any pixel belonging to a homogeneous region should be equal to

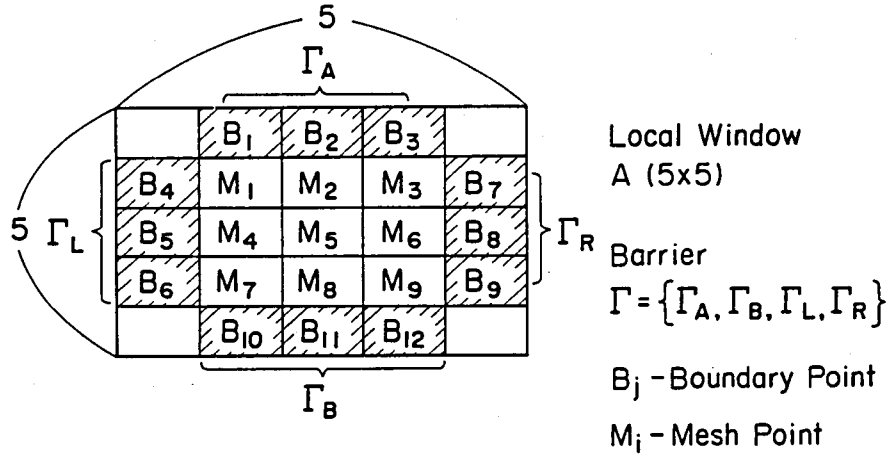


Figure 2-26 Local Window for Random Walk

1/4. The probability of leaving the barrier is 0 and the probabilities of leaving any mesh point are strictly positive. In the case of non-homogeneous regions, the moving probabilities are defined so that any particle will more likely move toward the direction of the smaller ascent or descent in the picture function. Therefore, such probabilities will be determined based on absolute differences in the picture function, or the gray scale value.

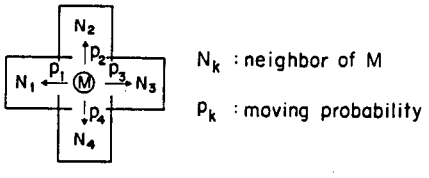


Figure 2-27 Moving Probabilities from Mesh Point to its Neighbors

The probabilities of moving left, up, right, and down from a given mesh point M to its corresponding neighbors $N_k(M)$ are denoted by $P_k(M)$, k = 1, 2, 3, and 4, respectively, as depicted in Fig. 2-27. They can be computed as follows:

$$\begin{cases} \sum_{k=1}^{4} p_k(M) = 1.0 \\ p_1(M) : p_2(M) : p_3(M) : p_4(M) = \frac{1}{d_1} : \frac{1}{d_2} : \frac{1}{d_3} : \frac{1}{d_4} \end{cases} \tag{2-25}$$

where $d_k = |f(M) - f(N_k(M))| + 1$. $f(M)$ and $f(N_k(M))$ stand for the picture values at pixel M and its neighbor $N_k(M)$, respectively.

(2) Degree of Non-Homogeneity

The degree of non-homogeneity for a given local window A can be derived by measuring the difference between distribution of the hitting probabilities over the barrier Γ in a homogeneous window (i.e., perfectly smooth window) and that in the window A.

Let us define the hitting probability $P(M_i, B_j)$ as the probability that a random walk started at mesh point M_i will end at boundary point B_j. The hitting probability for mesh point M_i can be found by solving the following system of linear equations:

$$\begin{cases} P(M_i, B_j) = \sum_{k=1}^{4} p_k(M_i)\, p(N_k(M_i), B_j) & \text{(i)} \\ \sum_{j=1}^{12} P(M_i, B_j) = 1.0 & \text{(ii)} \end{cases} \tag{2-26}$$

where $p(N_k(M_i), B_j)$ is also the probability of leaving mesh point $N_k(M_i)$, neighbor of M_i, and ending at boundary point B_j. If neighbor $N_k(M_i)$ falls on the boundary point, $p(N_k(M_i), B_j)$ is set to 0 or 1:

$$\begin{cases} p(N_k(M_i), B_j) = 1 \quad ; \quad \text{if } N_k(M_i) = B_j \\ \qquad\qquad\quad = 0 \quad ; \quad \text{otherwise .} \end{cases} \tag{2-27}$$

For a local window A of size 5x5, there are 108 equations of type (i) in Eq. (2-26), and 9 equations of type (ii). A solution of

Eq. (2-26) with conditions Eq. (2-27) can provide the desired hitting probabilities.

Another way to obtain the hitting probabilities is to actually simulate the planar random walk in the window of Fig. 2-26. As mentioned before, the probabilities of leaving a mesh point M_i for one of its 4 neighbors can define four disjoint intervals over $[0, 1]$ so that lengths of these intervals are proportional to the corresponding probabilities. A random number is picked up and then a particle moves in that direction as defined by the interval the random number belongs to. Suppose that a certain number n of particles leave a mesh point M_i and n_j particles will reach a boundary point B_j by the planar random walk simulation. If n is large enough in the statistical sense, n_j/n will be equivalent to $P(M_i, B_j)$, one of the hitting probabilities. Once the hitting distributions are known, the problem of texture discrimination can be solved by the following procedures.

Let the hitting distribution of a mesh point M_i be $(f_o)_i$ for a given window A and $(f_e)_i$ for the homogeneous window (i.e., the moving probabilities of any mesh point in a window are equal). The two distributions can be compared using the chi-square test[58]. There are four groups of 3 boundary points ($\Gamma_A, \Gamma_B, \Gamma_L, \Gamma_R$) and therefore 2 degrees of freedom for each boundary set Γ_K. The critical value is 5.89 at a 5% level of confidence. The "observed" distribution $(f_o)_i$ and the "expected" distribution $(f_e)_i$ are significantly different, if and only if

$$\sum_{j \in \Gamma_k} [(f_e^j)_i - (f_o^j)_i]^2 / (f_e^j)_i \geq 5.89 \qquad (2\text{-}28)$$

where $(f_o^j)_i$ and $(f_e^i)_i$ are the observed and expected frequency of hitting boundary point B_j when the random wlak starts at M_i. There are 9 mesh points in the window and the chi-square test is applied to each of them. A counter keeps track of the number of

times the expected and observed distributions differ significantly on boundary set Γ_k in the sense of chi-square test. The counter can be any of the values 0, 1, 2, $\cdots$, and 9. If the counter is equal to 9, a window of interest is completely different in comparison with the homogeneous window. If the counter is 0, a window is completely similar to the homogeneous one. It means that the counter number indicates the degree of similarity between test window and homogeneous one. This counting procedure is applied to all four boundary sets Γ_A, Γ_B, Γ_L, and Γ_R, whose counter numbers compose a four dimensional feature vector.

A horizontal and vertical raster scan over a given picture defines N windows A_i, i = 1, 2, $\cdots$ N, where the size of window A_i is 5x5 as described above. N depends upon the size of input picture and the scanning distance between two adjacent windows.

The feature vector is defined as the sum of differences in the hitting distributions for boundary sets Γ_A, Γ_B, Γ_L, and Γ_R between an input picture and a homogeneous picture. The feature vector could have been chosen to be any dimensional for each boundary point at no additional cost, and the decision to select a four dimensional feature vector was just one of convenience. If the texture discrimination problem is to assign a given picture to a class member in a given set of possible textures, then the closest neighbor classification based on the texture feature and a suitable metric will solve the problem.

The texture discrimination experiment was carried out on a PDP-11/45 minicomputer system.

The hitting distributions were obtained using a random number generation program and repeating the random walks: 50 times for each mesh point. These hitting distributions were compared with those obtained by solving the mathematical linear equations (2-26). No significant differences in the chi-square sense were

observed. The chi-square test would take into consideration only for those points in the boundary where the corresponding hitting distribution is above 5 % of the number of random walks performed. This provision assumes that small values do not lead to spurious results[58].

A set of input pictures for the texture discrimination are from Brodatz[59] and shown in Fig. 2-28. Fourteen texture patterns were digitized into a 128x128 picture. Each input texture was further divided into 4 frames of 64x64 size, yielding a test set for the texture discrimination. Therefore, the data set consists of 14 distinct classes and each class has 4 members. A set of feature vectors were derived for each of the 56 frames as described above. A classification by the nearest neighbor and a leave-one-out test[60] were used to check the discrimination performance. The leave-one-out test, suitable for small data sets, assigns a given texture pattern to the closest class as defined by the remaining 55 frames. There are 14 classes and the whole procedure is repeated 56 times. The performance represents the rate of correct assignments.

The results show that the performance improves if d (distance between two consecutive windows to test the random walk) decreases.

When d = 5 (i.e., windows are not overlapping), the performance is 60 % correct assignment. For d = 3 and 2, the correct assignments are 75 % and 86 %, respectively. Table 2-2 gives the confusion matrix of the texture discrimination for the input patterns shown in Fig. 2-28 and for a distance d = 2. There are 8 misassignments.

The random walk approach, implemented on PDP-11/45, was a rather slow process. It took approximately 1 sec. to obtain the hitting distribution for a window A (5x5) when the random walk was simulated 50 times. The texture discrimination procedure took 11

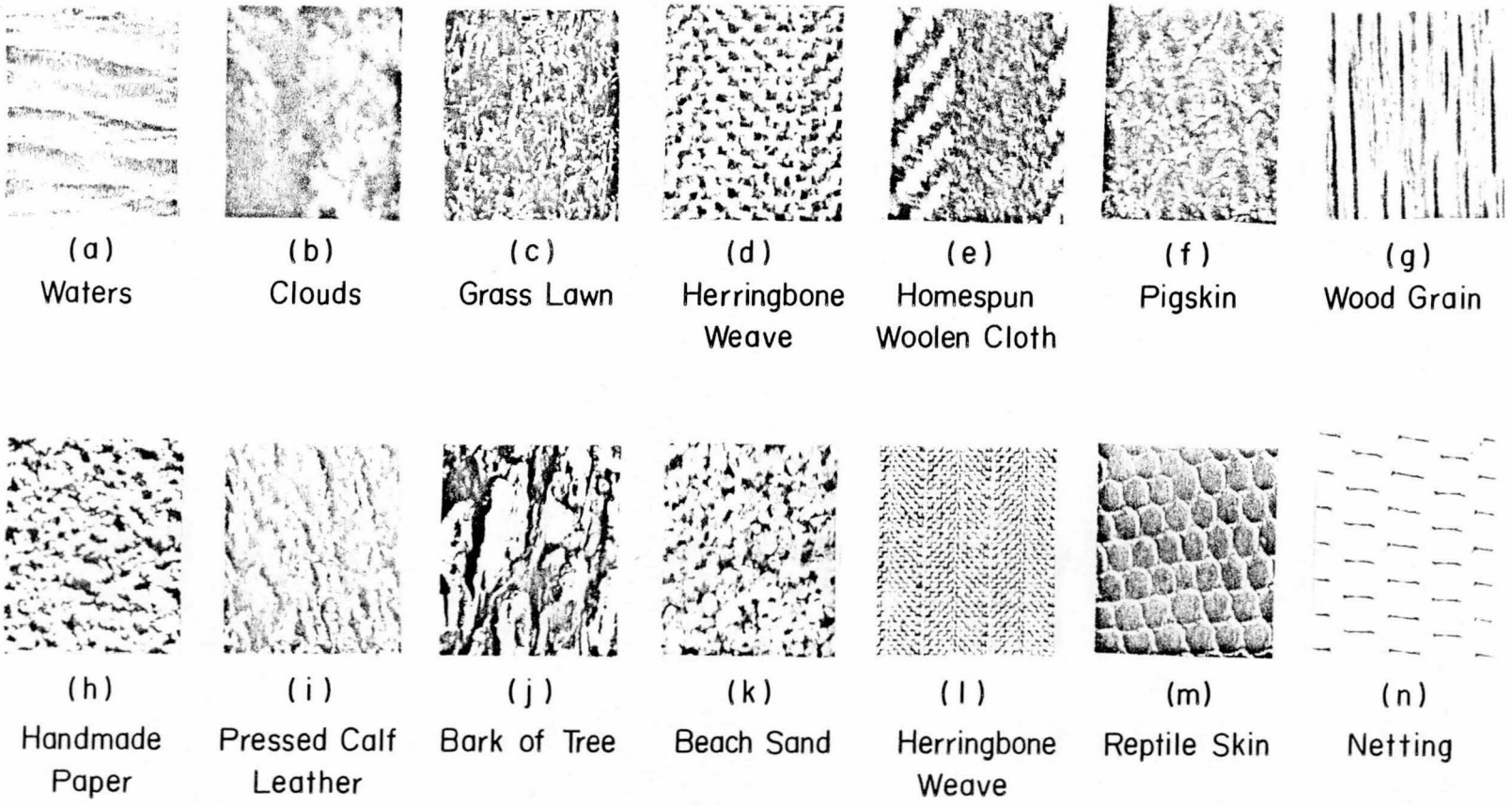


Figure 2-28 Texture Patterns from Brodatz's Photographic Album

	Pic. (a)	Pic. (b)	Pic. (c)	Pic. (d)	Pic. (e)	Pic. (f)	Pic. (g)	Pic. (h)	Pic. (i)	Pic. (j)	Pic. (k)	Pic. (l)	Pic. (m)	Pic. (n)
Pic. (a)	4													
Pic. (b)		1			①	①								①
Pic. (c)			4											
Pic. (d)				4										
Pic. (e)					4									
Pic. (f)					②	2								
Pic. (g)							4							
Pic. (h)								4						
Pic. (i)									4					
Pic. (j)									①	3				
Pic. (k)											4			
Pic. (l)					①							3		
Pic. (m)								①					3	
Pic. (n)														4

Table 2-2 Confusion Matrix Table of Texture Discrimination by Planar Random Walk Approach (○ indicates misassignments.)

minutes on frames of 64×64 and $d = 2$.

A comparison was performed between the random walk method and the usual texture discrimination based on the statistical approach. The features used were the Contrast and Angular Second Moment which are defined as follows:

$$\begin{cases} CON_1 = \sum_k k^2 D(k) \\ ASM_1 = \sum_k D^2(k) \end{cases} \tag{2-29}$$

$$\begin{cases} CON_2 = \sum_k \sum_\ell (k-\ell)^2 C(k,\ell) \\ ASM_2 = \sum_k \sum_\ell C^2(k,\ell) \end{cases} \tag{2-30}$$

where k and ℓ are gray level values of the picture, and D(k) and C(k, ℓ) are the probabilities of gray level difference and gray level cooccurence, respectively. The distance vectors (Δx, Δy) which define the above probabilities are (1, 1) and (-1, 1). There is no appropriate procedure available for selecting the features and distance vectors. Distance vectors studied here were chosen from the previous studies[18][19].

The two features and two distance vectors yield a four dimensional feature vector for each texture from (2-29) and (2-30). The same nearest neighbor classification and leave-one-out test were used as in the case of the random walk approach. The performances of classification test using the gray level difference and cooccurence were 79 % and 78 %, respectively, as compared with 86 % obtained by the planar random walk method.

58 項欠

CHAPTER 3 HARDWARE ARCHITECTURE OF LOCAL PARALLEL PICTURE PROCESSOR (PPP)

This chapter presents the hardware architecture of a newly developed picture processing machine.

Design concepts are first discussed in Section 3-1. Some technical points are considered on hardware realization: hardware architecture suitable for picture processing, most commonly used functions to be realized, speed, economy etc. Section 3-2 describes the hardware organization and functional modules. Detailed descriptions on the address control, pixel operations, and local operations are shown in Sections 3-3, 3-4, and 3-5, respectively.

Programming techniques for the machine and applications will be found in Chapter 4.

3-1 Design Concepts

Several picture processing machines have been proposed and some of them have already been realized. The idea of picture processing hardware has proved its feasibility and is demonstrating its capability.

The problem now to be solved is what architecture of picture processor is superior in terms of cost-effectiveness, or cost performance ratio. From trade-offs in speed, economy and flexibility, a microprogrammable local parallel picture processor (PPP) has been developed. PPP has employed the following ideas to improve

the cost-effectiveness in picture processing, which contribute to speeding up execution time and to reducing the hardware cost and complexity:

(1) Picture Memory Arrangement

Picture data is usually too large to fit into the main memory of the computer, and therefore a secondary storage, for example, disk or drum, must be used to contain the entire picture. As a result, there is a large amount of overhead "idle" time in transferring data between main memory and secondary storage for processing by the central computing unit. This data transfer time generally occupies a large fraction of the total throughput time in computer processing though it is not the essential calculations in picture processing.

In order to minimize the idle data transfer among memory hierachies, the picture memory has been designed. Fig. 3-1 shows an interactive picture processing system, comprised of the picture memory and PPP. Advanced memory fabrication tech-

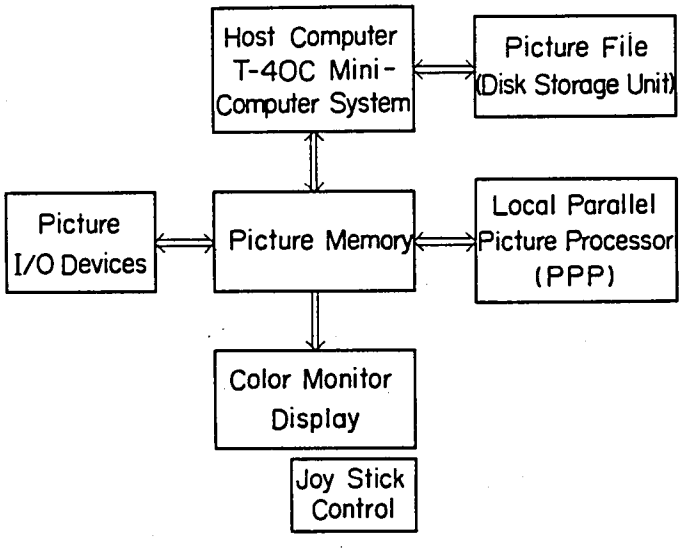


Figure 3-1 Interactive Picture Processing System

nology makes it possible to build the memory unit large enough to contain pictures.

PPP is directly connected to the picture memory unit, as illustrated in Fig. 3-1. The picture memory plays several important roles in the picture processing system, such as working memory for PPP, high speed buffer memory for I/O devices, refresh memory for color display, and external bulk memory for host computer.

(2) Selection of Important Picture Processing Functions

There are several types of picture processing functions, which are frequently used and take a large amount of execution time.

Sum-of-Products - Two Dimensional Convolution

A two dimensional convolution is defined on an input picture F(x,y) by

$$G(x,y) = \sum_i \sum_j W_{ij} F_{ij}(x,y) \qquad (3\text{-}1)$$

where F_{ij} (x, y) are neighboring pixels around (x, y) of an input picture and W_{ij} are weights. Depending upon the weights, this computation can be: two dimensional convolution used in averaging, enhancement and differentiation; correlation or matching used in object identification etc.

Point Mapping - Data Conversion

A data conversion is defined by

$$G(x,y) = \phi\{F(x,y)\} \qquad (3\text{-}2)$$

where ϕ is a single-valued (usually nonlinear) function. Point mappings are used for such operations as contrast stretching, gray scale mapping, nonlinear filtering, multilevel thresholding, pseudo-color generation, intensity correction of sensors and displays etc.

Linear Coordinate Transformation - Affine Transformation

An affine transformation is specified by a linear coordinate

transformation matrix and shift values as follows:

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} \tag{3-3}$$

where (X, Y) and (x, y) are the output and input coordinates, respectively. This operation is used in magnification, shrinking, rotation, shifting etc.

Fast Transforms - Fast Fourier Transform (FFT)

A fast transform is given by

$$F(u,v) = \sum_x \sum_y F(x,y)\, W^{ux+vy} \tag{3-4}$$

where W is the kernel of the transform. The most commonly used fast transform is the Fourier (kernel $W = \exp(-jz)$). Fast Fourier transform has been used in restoration, enhancement, reconstruction, coding, compression etc.

Logical Filtering

A logical filtering is defined on an input binary picture B(x,y) by

$$C(x,y) = L_f\{\sum_i \sum_j B_{ij}(x,y)\} \tag{3-5}$$

where L_f is a mapping function specified by a certain matrix of an input picture. Logical filtering operation is used in boundary detection, thinning, swelling, line segment coding, feature point extraction etc.

Pixelwise Arithmetic and Logical Operations

Pixelwise operations are defined on more than two input pictures by

$$\begin{aligned} G(x,y) &= A\{F_1(x,y),\ F_2(x,y) \cdots\} \\ C(x,y) &= L\{B_1(x,y),\ B_2(x,y) \cdots\} \end{aligned} \tag{3-6}$$

where A and L are an arithmetic operation and a Boolean logical operation, respectively. These operations are used in vignetting correction by original and shading data, other color information calculation (hue, saturation and brightness) by red,

green and blue data, multi spectral data processing (ratioing, normalization) etc.

Histogram Computation

The histogram computation is performed as follows:

$$\begin{cases} g = F(x,y) \\ H(g) \leftarrow H(g) + 1 \end{cases} \tag{3-7}$$

where g is a gray level data of an input picture F(x,y) and H is a buffer for histogram data. Histogram computation is commonly used in frequency measurement of gray levels, gray level statistics, area counting etc.

Miscellanea

Region labeling, arc length and slope detection, and graph representation of a line drawing are commonly used as feature extractions from binary pictures.

Projections and cross sections are also used for gray level pictures.

From points of view in hardware simplicity and functional requirements, the following basic picture processing functions have been chosen to be implemented in PPP: two dimensional convolution, logical filtering, region labeling, affine transformation, data conversion, histogram computation, and pixelwise operations.

From lack of the bit length in PPP, hardware implementation of fast Fourier transform was excluded. Other inherently sequential operations were also rejected.

(3) Local Parallel Implementation

There are two types of hardware implementation of parallelism in picture processing. The most intuitive idea is a fully parallel array of identical processing elements corresponding to each pixel, which work simultaneously on picture data. However, picture processing machines of this type have big problems in practical assembly and processing capability. Depending upon the

size of pictures to be processed, a fully parallel machine must have a huge amount of circuit cells. For example, CLIP 4 has been realized by LSI technology. It is still only possible to place 8 cells of CLIP 4 processor on one LSI chip. Thus, it requires a number of LSI chips and associated wirings. Another problem is data communication among pixels. A fully parallel machine usually restricts communication only between adjacent neighbors. This limitation makes it difficult to implement picture processings of Eqs. (3-1) and (3-3), or results in excessive time to access other neighbors, which cancels the speed-up effects brought about by a fully parallel basis. The communication range of neighboring pixels depends upon the nature of picture operations, but it usually falls between 3x3 and 16x16 pixels surrounding the central pixel.

These considerations have led PPP to carry out picture processings on a local parallel basis. Specially designed circuits accomplish the local parallel processing: i.e., parallel data access and computation in the local window. Then, the center position of local parallel operation is shifted sequentially to the next position as in a raster scan. The combination of local parallel operation and sequential scanning fits the serial data transfer characteristic of current I/O devices and picture memory, and also reduces the hardware cost and complexity.

(4) Flexible Control Structure

Even though several basic picture processing functions are realized in PPP, it is still required to perform more global and complex functions, such as texture analysis, shape identification, region separation and so forth.

PPP has a microprogram architecture to accomplish more flexible operations by combining basic functions or using an included microprocessor.

(5) Control Structure for Local Parallel Operation

As indicated in Chapters 1 and 2, local parallel picture processing consists of program loop control, data access and computation procedures. PPP controls each procedure independently, which results in high speed processing. Specially designed circuits are prepared for parallel data access and parallel computation of local operations. A pipeline control technique also helps PPP to speed up local operations.

3-2 Hardware Architecture

The microprogrammable local parallel picture processor PPP has been implemented under design concepts described in Section 3-1. Fig. 3-2 depicts an overall blockdiagram.

The host computer is a minicomputer TOSBAC-40 Model C with 64 K bytes main memory. The host computer controls operational modes of PPP and picture memory under an interactive operating software system. Picture processing programs which use PPP will assign the picture memory to input and output, set parameters in PPP, specify the basic function, issue a start command, and then sense the end status of execution from PPP. Another important role of the host computer is to transfer microprograms to PPP. Details on these software techniques will be discussed in Chapter 4. The interface to the host computer controls to transfer microprograms and commands with a set of parameters, and to generate an interrupt signal at the end of each operation.

The picture memory consists of four layers of 512 x 512 pixels with 8 bits per pixel and four planes of 512 x 512 binary graphic

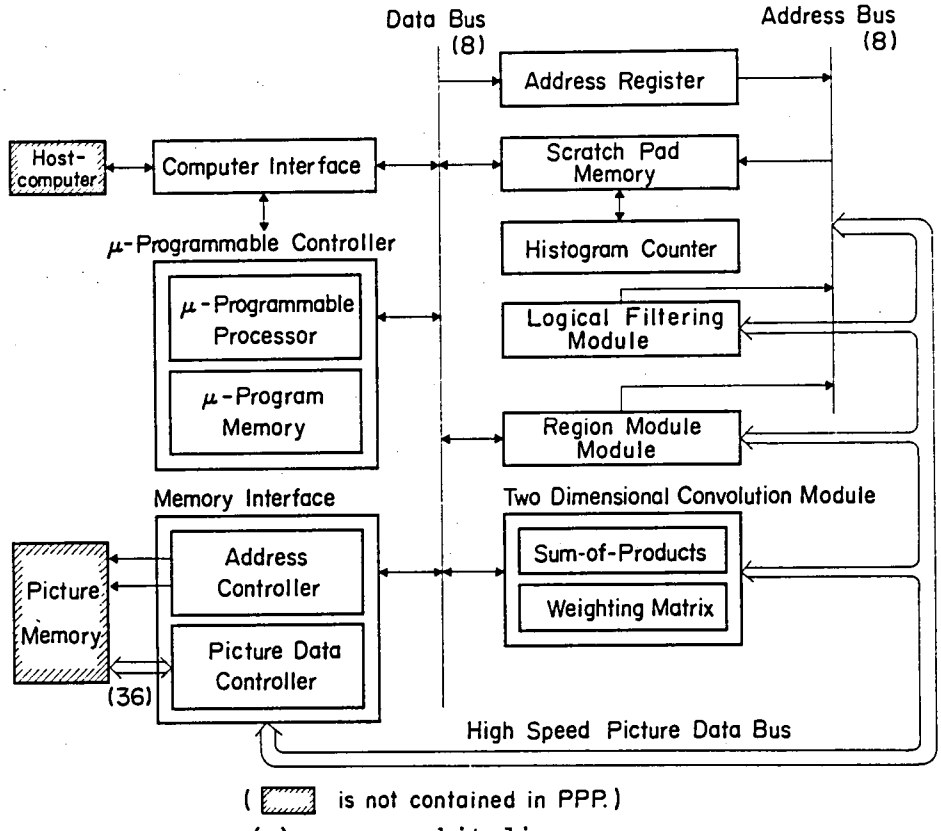


Figure 3-2 Overall Blockdiagram of PPP

memory. The picture memory is directly connected to the high speed picture data bus in PPP, which has a transfer rate of 1 M words (36 bits per word) per second. The interface to the picture memory has the address controller and data controller. The address controller calculates the input and output addresses independently, and the data controller transfers picture data simultaneously to or from any layer or plane of the picture memory. The address controller is capable of calculating next addresses automatically and checking the boundary of picture data to be processed.

The linear coordinate transformation of picture data, for instance, takes a great advantage of these capabilities of the address controller. Details on the address control can be seen in

the next section.

The interface to the picture memory also performs the input/output data control. The 36-bit data at a given pixel address in the picture memory are fed through the data controller to the internal data bus or high speed picture data bus. One of the four 8-bit gray scale data, or binary data out of 4 graphic data can be selectively presented to the data bus under the microprogram control. On the other hand, the output data through the internal data bus is bufferred at the proper bit position in a 36-bit register under the microprogram command. The host computer specifies layers of the picture memory to accept the corresponding output data from PPP prior to the data transfer in executing picture processings.

Modules for basic picture processing functions are specially designed: two dimensional convolution, logical filtering, region labeling, data conversion, and histogram computation.

The two dimensional convolution module has a buffer memory of shift registers and a weighting matrix memory. The shift register stores picture data of 7 lines and therefore a parallel accessing of an 8x8 local window data is possible. A sophisticated parallel calculation circuit to obtain sum-of-products has been designed with a pipeline technique (See Section 3-5).

The logical filtering and region labeling modules also have a buffer memory to form a local window of 3x3 and 3x2 binary picture data, respectively. The region labeling module includes a new number counter to label each component and two sets of read only memory to control the connectivity (4-neighbor or 8-neighbor).

The scratch pad memory consists of 512 bytes, which is used for table look-up operations in logical filtering, data conversion, and region labeling. It is also used as 256 registers of 16-bit length in histogram computation. The histogram counter is used for incrementing the content of histogram buffer, that is, the scratch pad memory. Data transfer from or to the scratch pad

memory is done through the interface to or from the host computer.

The microprogrammable controller (μC) plays an important role in PPP. Its functions are as follows: communication to the host computer, interface control to the picture memory, internal data bus control, control of each functional module, and pixelwise operations. The μC consists of the microprocessor with arithmetic and logic unit (ALU), multiplier, and the microprogram memory. Microprograms for basic functions can be stored in the read only memory, while any user's microprograms can be transferred from the host computer to the random access microprogram memory for execution. The word length of a microprogram instruction is 40 bits, whose cycle time is 250 nanoseconds.

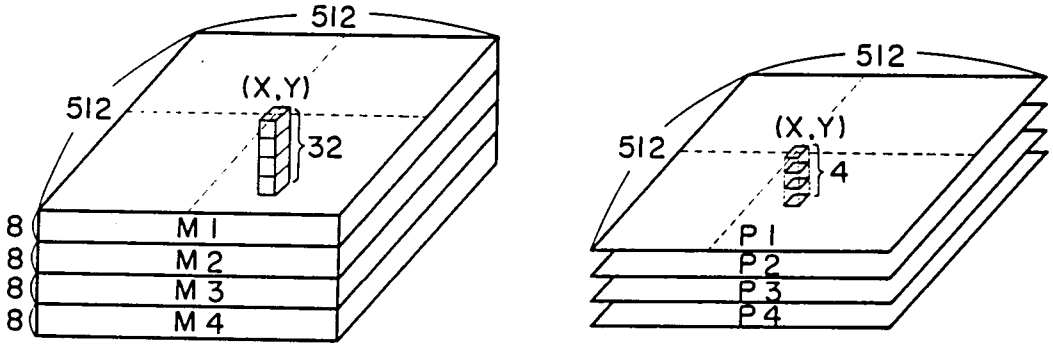
Design concepts listed in the previous section have been fulfilled as follows:

(1) Picture Memory Arrangement

The picture memory is the primary component of the interactive picture processing system as shown in Fig. 3-1. It has been built so that it can contain four gray scale pictures of 512 x 512 pixels and four binary pictures of the same size. These numbers have been chosen mainly for applications in remote sensing.

Fig. 3-3 illustrates the picture memory organization. The image memory (gray scale picture memory) is composed of four 256K byte solid-state random access memory (RAM) banks (designated as M1, M2, M3 and M4) and the plane memory of four 256 K bit RAM banks (P1, P2, P3 and P4). The picture memory consists of 4 K dynamic RAM chips.

The image memory can be used for storing four spectral bands of remotely sensed pictures (for example, Blue, Green, Red and InfraRed data), or three color pictures (B, G and R). The



(a) Gray Scale Picture Memory (Image Memory)

(b) Binary Graphic Memory (Plane Memory)

Figure 3-3 Picture Memory Unit

plane memory is used for graphic picture data such as line drawing, object area or boundary, and characters. These picture data can be monitored on the color display, and four graphic planes are overlayed by the prespecified color information: P1 to white, P2 to red, P3 to green, and P4 to blue.

PPP is directly plugged into the picture memory. PPP can get picture data every 1 μS from any layer or plane of the picture memory and output resultant data to any location in the memory of the calculated address by the interface.

(2) Selection of Basic Picture Processing Functions

Local parallel operations have been contrived in individual functional modules: two dimensional convolution, logical filtering, region labeling, linear coordinate transformation, data conversion, histogram computation, and microprogrammed pixelwise operations. Table look-up technique which is commonly used in logical filtering, data conversion and histogram computation have been implemented by employing the scratch pad memory. Region labeling also makes use of this scratch pad memory as a buffer to store intermediate labeled number.

Table 3-1 shows a list of a repertoire of picture processings

where the PPP's basic functions can be applied.

Basic Function	Applications
Two-Dimensional Convolution	Smoothing, Noise averaging, Enhancement, Differentiation, Restoration
Affine Coordinate Transform	Enlargement, Shrinking, Rotation, Shift Geometric Correction, Mosaicking
Logical Filter	Thinning, Noise Clearing, Gap Filling, Feature Detection
Region Labeling	Region Separation, Region Coloring, Area Counting
Data Conversion	γ-correction, log/exp, sin/cos, $x^2/\sqrt{x}$ Thresholding, Histogram Equalization
Histogram Generation	Gray Level Frequency Counting, Area Counting
Pixel Operation	Pixelwise Arithmetic and Logical Operations, Ratioing, Shading Correction

Table 3-1 Applicable Picture Processing Repertoire

(3) Local Parallel Processor

PPP has been realized by combination of local parallel operations and their sequential scanning. Local parallel operations in two dimensional convolution, logical filtering and region labeling can be accomplished by preparing line buffers for parallel data access, and specially designed circuits for parallel computation in a local window.

Especially in the two dimensional convolution module, a very

sophisticated circuit technology permits PPP to perform rapid computation of the sum-of-products and to reduce the number of multipliers and adders.

(4) Flexible Processor

PPP has not only special circuits for basic functions but also a microprogrammable controller for flexible operations.

The microprogrammable controller includes the bit-slice microprocessor and multiplier. The microprocessor executes several microinstructions of an arithmetic and logic unit (ALU) to perform arithmetic, logical and shifting operations on 20-bit data.* There are general-purpose registers with two parallel address capabilities. The multiplier works on 8-bit integers. The microprogram memory is reserved for user's programs to combine basic functions or to perform the microprocessor operations. The microprogram architecture would permit great flexibility to achieve high speed picture processing with PPP.

(5) Control Structure of Local Parallel Operation

As mentioned in Section 3-1, there are three separable procedures in the local parallel operation: program loop, data access, and computation. Each procedure has been hardwired for the high speed processing capability. The loop control is achieved by the address controller. The interface to picture memory allows the simultaneous data accessing of input and output picture, and each module is equipped with line buffers to enhance the parallel data accessing capability of the local window. The computation in the local window is also refined to improve the speed and to reduce the hardware complexity.

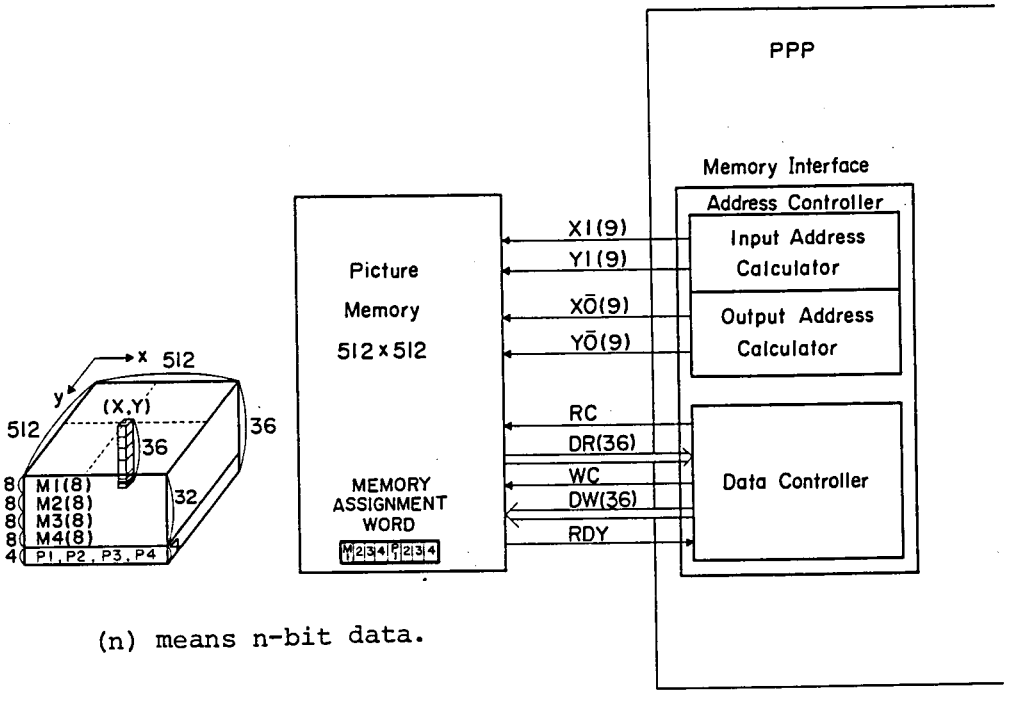
* See the bit specification in Section 3-3 (p. 77).

3-3 Address Controls in PPP

The address control plays an important role in local parallel picture processing machines. Not only picture data access but also program loop control are attained by the address controller.

The interface to the picture memory in PPP is responsible for the independent calculation of input and output addresses, the boundary check of the picture, and the address computation of the linear coordinate transformation. The address controller consists of input and output address calculators as shown in Fig. 3-4.

The interface executes high speed data accessing as the input and output addresses are fed through (Fig. 3-4). The interface is provided with dual 18-bit address lines, dual 36-bit data lines (DRs and DWs), and three control lines. 36-bit data (four 8-bit gray scale and four binary data of each pixel) is transmitted from and to the picture memory with independent 18-bit address data and corresponding control signals (RC, WC, and RDY).



(n) means n-bit data.

Figure 3-4 Picture Memory and Interface

18-bit address data consists of two 9-bit addresses for x and y coordinates in the two dimensional picture. Each 9-bit address data is independently calculated by a high speed arithmetic unit together with registers included in the microprocessor and up/down counters, for input and output address calculators, respectively. Each register or counter is initially loaded an appropriate value from the host computer at the beginning of picture processing.

There are two major address control mechanisms in PPP: the address control for the local operation and for the coordinate transformation.

(1) Address Control of Local Operation

The local operation is usually performed by the raster scan, line scan from left to right and then from top to bottom. Some local operations, however, are sensitive to the scanning direction: for instance, a thinning operation. Eight possibilities can be considered in the raster scan as indicated in Fig. 3-5.

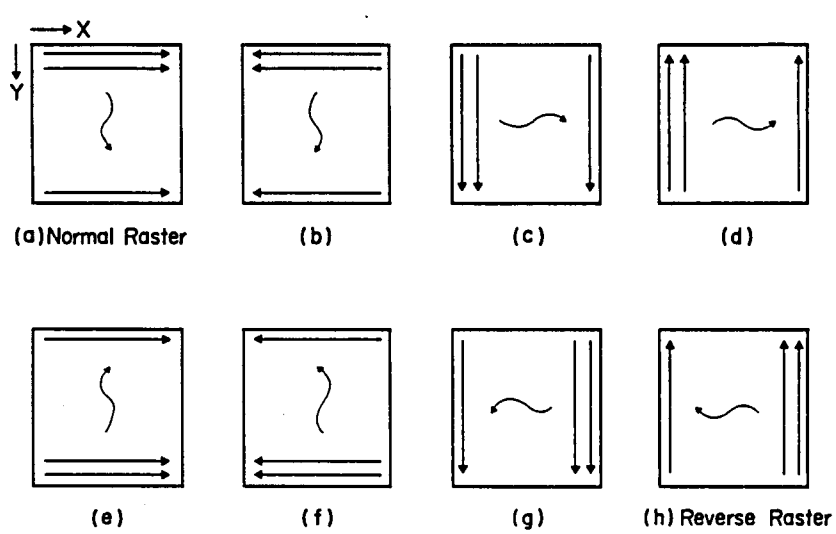


Figure 3-5 Raster Scanning Modes

Address calculation of the input and output picture specified by one of the eight raster scan modes, can be implemented by the up/down counter. Fig. 3-6 shows the implementation of the output address calculator.

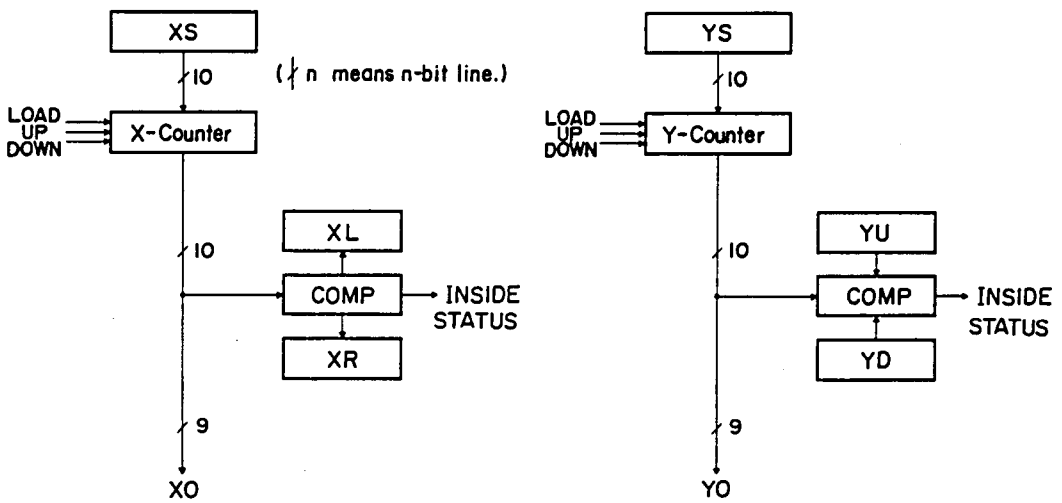


Figure 3-6 Output Address Calculator

Depending upon the selected mode of scan, the up/down control signal and initial values are properly given. For example, in the case of the normal raster scan the X-counter is first set by LOAD signal to a given starting address (XS) at the beginning of each line scan and then counted up during the line scan, while the Y-counter is also initiated to be a certain value (YS) and then counted up at the completion of each line scan. The reversed raster scan is done by alternating the up/down control signal and initial address values. The column scan is done from bottom to top and then from right to left.

In the case of local picture operations, the input address calculator works in the same manner as the output one does. Implementation of the input address calculator will be discussed in the next section in connection with the coordinate transformation.

(2) Address Control in Linear Coordinate Transformation

The coordinate transformation is a space-to-space mapping operation that requires resampling. Let us consider the following example:

When digitized aerial photos are taken at two different times over the same area, it is often required to rotate, translate and scale one picture in order to obtain the conformity of two pictures and detect the changes. The necessary transformation (rotation, translation and scaling), can be written in the form of Eq. (3-3). The resampling process is required for finding the value of sample points in the mapped output picture. In Fig. 3-7, the sampling grid pattern is defined by X and Y coordinates, and resampling occurs at the intersections of solid lines. An attempt to interpolate the value at any resampling grid point of the output picture requires the use of all sampled values in the input picture. Since the interpolation algorithm of this type imposes severe computational demands, three approximate interpolation algorithms are generally used in the input domain: nearest neighbor, linear interpolation, and cubic convolution.

It is desired here to find the value at the circled point, for instance, (X_4, Y_4) of the output picture as shown in Fig. 3-7. The F_1 value is known for the input grid point (x_1, y_1), and the F_2, F_3, ..., F_{16} values are also known. The distances between the circled point and each input grid point can be calculated. The value $G(X_4, Y_4)$ is given by the following

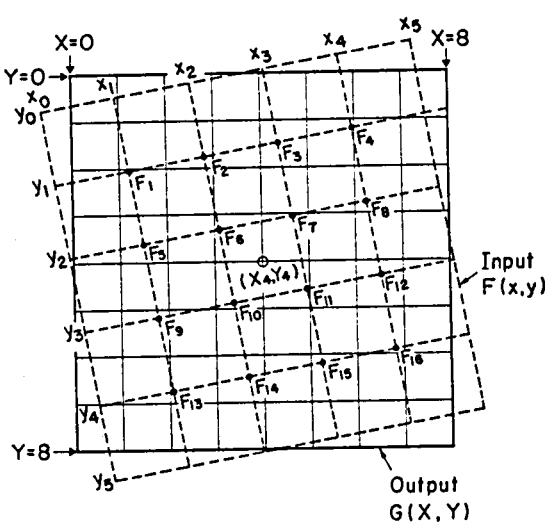


Figure 3-7 Linear Coordinate Transformation from Input Space (x,y) to Output Space (X,Y)

equations. Subscripts N, L, and C of G(X, Y) indicate the value obtained by the nearest neighbor, linear interpolation, and cubic convolution, respectively.

$$G_N(X_4, Y_4) = F_{10} \tag{3-8}$$

where the grid point (x_2, y_3) is the nearest to (X_4, Y_4).

$$G_L(X,Y) = (1-\Delta Y)\{(1-\Delta X)F_6 + \Delta X \cdot F_7\} + \Delta Y\{(1-\Delta X)F_{10} + \Delta X \cdot F_{11}\} \tag{3-9}$$

where ΔX and ΔY are the distances between (X_4, Y_4) and (x_2, y_2) in each direction.

$$G_C(X_4, Y_4) = \sum_{i=1}^{16} W_i F_i \tag{3-10}$$

where the weights W_i are given by the sinc function of the distance between (X_4, Y_4) and interestections of dotted lines.

As Fig. 3-7 implies, the resampling process of the output picture can be accomplished at each point by computing neighbor grid positions (x_i, y_i) of the input picture. Eq. (3-3) can be rewritten to calculate the input grid position (x, y) for a given resampling point (X, Y) as follows:

$$\begin{cases} x = pX + qY + x_0 \\ y = rX + sY + y_0 \end{cases} \tag{3-11}$$

If the resampling point varies according to the normal raster scan in the output picture, the input positional value (x, y) can be obtained very quickly with the aid of the high speed arithmetic unit. Fig. 3-8 indicates the input address calculator implemented by the included ALU and registers. Coefficients p and r in Eq. (3-11) are set in the registers DXX and DXY; x is incremented by the

amount in DXX when X is counted up by one and Y remains constant within each line scan. Coefficients q and r, equal to the contents of DYX and DYY registers respectively, provide the increment at each completion of line scans. The offset values are x_0 and y_0 set in the initial address registers XS and YS.

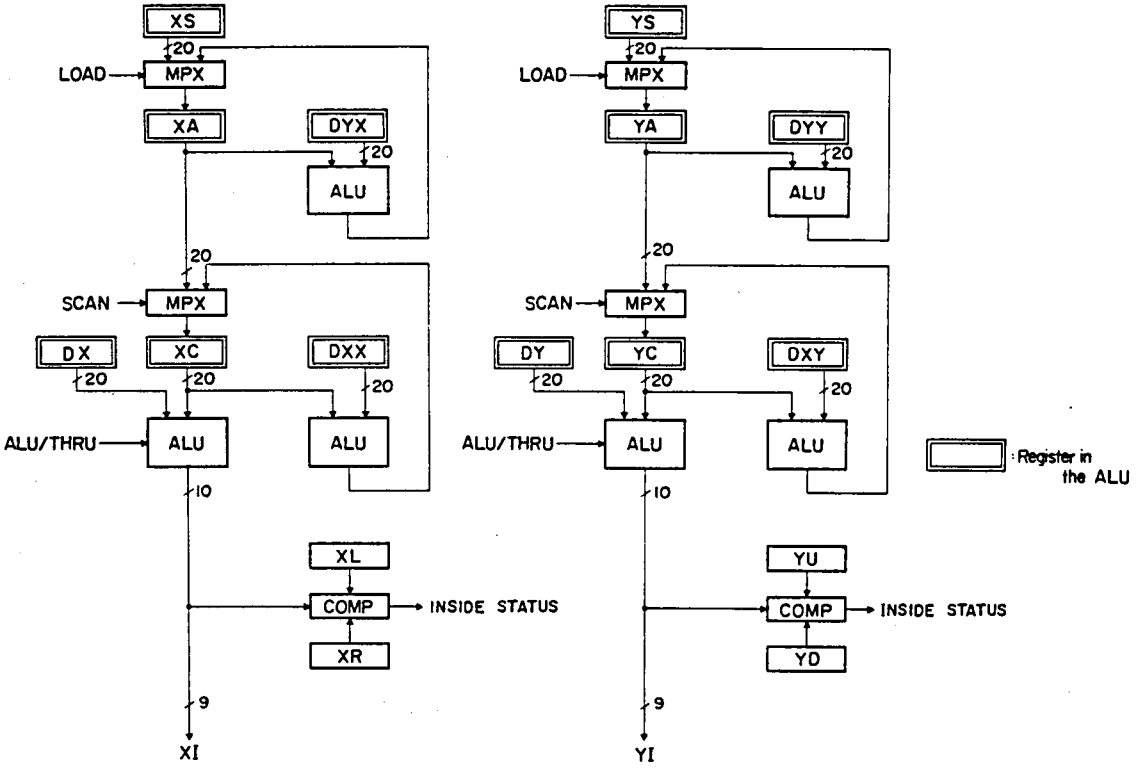


Figure 3-8 Input Address Calculator

⟊n indicates n-bit lines.
XL,XU,YL, and YU are the boundary information to be processed.

All values in the input address calculator are expressed in 20 bits with 9-bit fraction. The number of bits of address data has been determined so that a picture might be rotated with positional accuracy of 1/512 (512 is the maximum picture size in the memory). Since the computed values of x and y are not always an integer,

the interpolation for finding the value of output sample points is activated. The address calculator in Fig. 3-8 has an additional register DX or DY to compute the addresses of neighboring grid points in the input picture at high speed. If the registers DX and DY are both set to 0.5, the input address controller is dedicated to access the nearest neighbor pixel. When the linear interpolation algorithm is used at the resampling process, DX and DY registers will be set to (0.0, 0.0), (0.0, 1.0), (1.0, 0.0), and (1.0, 1.0) in order to sequentially access four adjacent neighbors: i.e., F_6, F_{10}, F_7, and F_{11} in Fig. 3-7. Similarly, the registers DX and DY enable the address controller to access sixteen neighboring points ($F_1 \sim F_{16}$) by changing the contents from -1.0, 0.0, 1.0 through 2.0. The value 0.5 of these additional registers raises fractions not lower than 0.5 to unity for effective integer address data, while values 0.0 and 1.0 mean the unconditional cut-off and raise of fractions, respectively.

Implementations of the input and output address calculators by the ALU and counter, respectively, emphasizes the flexible address control both for local operations and coordinate transformation. In addition, the address controller can specify the rectangle boundary to be processed as shown in Fig. 3-9. Each address calculator in Figs. 3-6 and 3-8 generates an inside status signal by comparing the current address with the preset boundary address data (XL, XR, YU, and YD). One of status

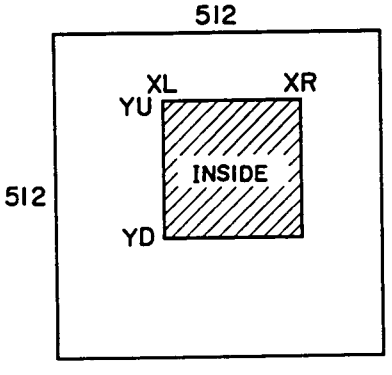


Figure 3-9 Rectangle Boundary for Processing

signals from four address calculators (XI, YI, XO, and YO) is predetermined to create the scan-line-end and frame-end status, which are sensed by the microprogram.

The address controller, however, is free from the address computation of pixels in a local window. In an attempt to access picture data of the local window in parallel, a set of line buffers (shift registers) is equipped for each local parallel operation of the two dimensional convolution, logical filtering and region labeling. A set of line buffers forms a local window to present their data to the computational circuit in parallel. The size of local windows in the two dimensional convolution, logical filtering and region labeling, are 8x8, 3x3 and 3x2, respectively, which will be described in Section 3-5.

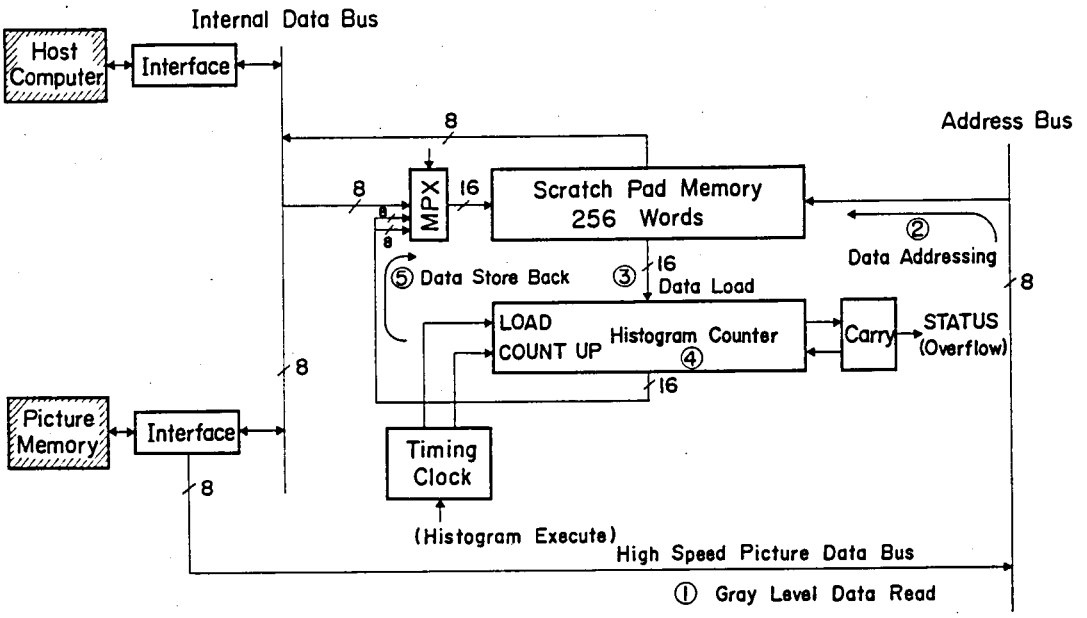
3-4 Pixel Operations

In what follows, by a pixel operation is meant a process which handles data pixel by pixel or a function whose values at an output pixel depend only on the corresponding pixel in the input picture.

PPP has the capability of executing three kinds of pixel operations at high speed: the histogram computation, data conversion (point mapping), and microprogrammed pixelwise operations. These pixel operations are usually performed in the raster scan mode.

(1) Histogram Computation

A histogram represents the number of pixels versus the gray level value. With reference to Fig. 3-10, the histogram computation



① indicates the i-th operational sequence.

Figure 3-10 Histogram Computation Mechanism

is performed by means of the 256-word scratch pad memory and an associated counter.

The raster scan locates each pixel of the input picture, reads its gray level through the interface from the memory (①), and the content of the scratch pad memory, whose address is equal to that gray level (②), is loaded into the histogram counter (③). The counter is incremented by one (④) and then its content is stored back into the scratch pad memory with the same address (⑤). The sequence of the histogram computation is repeated for a predetermined input data area. The histogram computation is executed in 1 μs for each pixel.

Since the bit length of the scratch pad memory is limited to 16 bits, the histogram computation is valid only for counting up to $2^{16}-1$ pixels. This means that the histogram computation can be applied to a 256x256 picture of a single gray level. However, it can usually be applied to 512x512 picture data because they have

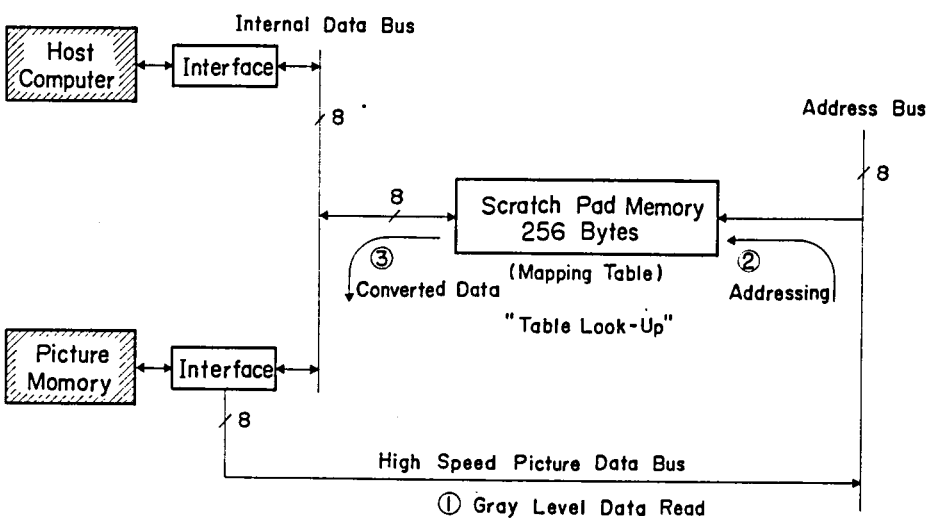
various gray levels. If the histogram counter overflows, the carry signals the corresponding status to the host computer, and the count is fixed to the highest value.

After completion of the histogram computation, the content of the scratch pad memory, which represents the frequency counting of gray levels, will be transferred via the interface to the host computer under a program control.

(2) Data Conversion (Point Mapping)

With the scratch pad memory, the data conversion function of Eq. (3-2) can be achieved in a simple manner.

Fig. 3-11 illustrates the data conversion sequence. The gray level data from the picture memory via the interface (①) is presented to the address bus of the scratch pad memory (②). The content of the scratch pad memory is looked up according to the incoming gray level value and is output to the picture memory through the interface. Since each 8-bit byte data represents a unique scratch pad memory address, the scratch pad memory functions as a high



ⓘ indicates the i-th operational sequence.

Figure 3-11 Data Conversion Mechanism

speed data converter. An appropriate table of the mapping function in Eq. (3-2) is loaded at the beginning of the data conversion operation from the host computer.

Depending upon the table data in the scratch pad memory, the data conversion may carry out gray scale compression or expansion; nonlinear data mapping on each pixel; substitute the logarithm, antilogarithm, square, root, or any mathematical function of the pixel value; create a binary picture by thresholding input gray levels with a given value; and so forth.

The data conversion is also accomplished in 1 μS per pixel by the raster scan mode and the rectangle boundary to be processed is prespecified by the host computer.

(3) Microprogrammed Pixelwise Operations

As mentioned before, the picture memory has the capacity to store four gray scale pictures and four binary pictures of 512 x 512. There are several cases for processing multilayers of pictures: temporal picture comparison, color picture manipulation, multispectral data processing as follows:

Gray Level Correction

A picture input device ideally should map brightness into gray levels independently of position. Unfortunately, many optical systems suffer from vignetting (or shading), whereby the picture is attenuated more near the edge in comparison to the central portion.

The correction of gray levels to compensate this shading is given by the pixelwise subtraction, i.e.,

$$G(x,y) = F(x,y) - S(x,y) \qquad (3\text{-}12)$$

where F is an observed picture and S is a standard picture obtained by observing a uniformly bright plane through the same optical system.

Change Detection

Two preregistered temporal pictures are used to detect changes which have occurred during time interval: chest radiographs for clinical diagnosis, aerial photographs for land use management etc.

A simple change detection technique is also the pixelwise subtraction, i.e.,

$$C(x,y) = F_{t2}(x,y) - F_{t1}(x,y) \tag{3-13}$$

where F_{t1} and F_{t2} are two time consecutive pictures.

Color Picture Manipulation

A color picture is usually composed of three pictures corresponding to the tri-stimuli (red, green and blue). From these component pictures, the other color information can be estimated by arithmetic operations at each pixel: brightness, hue and saturation.

Multispectral Data Processing

Multiband aerial photographs are useful for various applications in remote sensing: crop identification, land cover classification, pollution detection, and so on. A multiband photo typically consists of four pictures which are filtered by the blue, green, red and infrared spectral bands.

Several techniques for multispectral data processing have been studied. For example, ratioing, difference-sum ratio, and normalization are applicable techniques done by the following pixelwise operations:

$$\text{Ratioing:} \quad R_1(x,y) = S_i(x,y) / S_j(x,y) \tag{3-14}$$

$$\text{Difference-Sum Ratio:} \quad R_2(x,y) = \frac{(S_i(x,y) - S_j(x,y))}{(S_i(x,y) + S_j(x,y))} \tag{3-15}$$

$$\text{Normalization:} \quad R_3(x,y) = \frac{S_i(x,y)}{\sum_{j=1}^{4} S_j(x,y)} \tag{3-16}$$

where S_i is the i-th picture in the multispectral pictures.

Classification of each pixel into one of given categories is an important processing. A common technique for this treats each pixel data as a four dimensional vector and applies classification algorithm (parallel-piped thresholding, maximum likelihood estimation).

The parallel-piped thresholding compares each vector component value with lower and upper limit values of a given category.

The maximum likelihood estimation computes the Mahalanobis distances between each vector and categories, and finds a category with the minimum distance. These operations can be carried out by the pixelwise operation.

Pixelwise Logical Operations

For example, two binary pictures are used to find a common portion. This manipulation is expressed by the logical AND operation (·) of two binary pictures B_1 and B_2.

$$C(x,y) = B_1(x,y) \cdot B_2(x,y) \tag{3-17}$$

Other Boolean logical operations available: OR, Exclusive OR etc.

PPP has been designed to carry out as many kinds of pixelwise operations as possible. For this purpose, microprogrammable controller is equipped with a high speed arithmetic and logic unit (ALU) and a multiplier, as shown in Fig. 3-12.

The ALU can execute three arithmetic operations (T+S, T-S, and S-T) and five logical operations (T OR S, T AND S, $\overline{T}$ AND S, T EXOR S, and T EXNOR S); T and S are paired-operand data at the ALU input. The shifting operation can also be performed

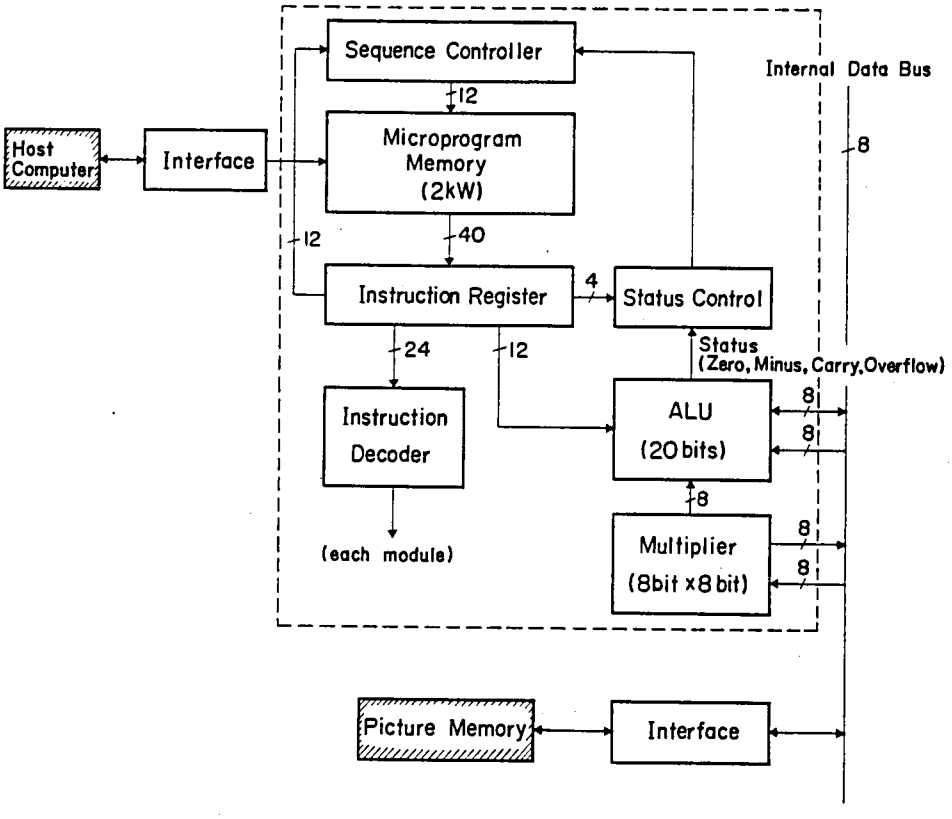


Figure 3-12 Microprogram Controller in PPP

in the ALU's register. The ALU operates on 20-bit data, which is also used in the input address calculator as described in the previous section.

A multiplier is connected to the internal data bus, which has two 8-bit input registers and a 16-bit product register.

The ALU and multiplier, in conjunction with the data bus, provide powerful pixelwise arithmetic and logic operations. PPP may carry out more complex pixelwise operations by combining basic functional modules. The ratioing of Eq. (3-14), for example, can be established by the logarithmic data conversion, subtraction, and then exponential data conversion (in the restricted form because of bit precision) under a microprogram control.

Any user's microprogram can be loaded from the host computer for execution. The execution time depends on the number of

microprogram steps per each pixel; one microprogram step can be performed in 250 nS.

3-5 Local Parallel Operations

The local parallel operation means a function, whose values defined at an output pixel, depend on a set of neighboring pixels (the local window) in the input picture (See Section 2-1).

PPP can perform three types of local parallel operations at high speed by specially designed circuits: the two dimensional convolution for gray scale pictures, logical filtering and region labeling for binary pictures. Notice that these local parallel operations are sequentially executed in the raster scan mode, while calculations or processing in the local window are done in parallel.

(1) Two Dimensional Convolution

Fig. 3-13 depicts an illustrative framework of the two dimensional convolution given by Eq. (3-1). When the size of weighting matrix is m x m, it includes m^2 multiplications and $m^2 - 1$ additions. A fully parallel implementation would require m^2 multipliers and $m^2 - 1$ adders, and would be much faster. However, even for a small weighting matrix (for instance, m = 8), a large number of parallel arithmetic elements would be necessary.

In PPP, sophisticated time-shared pipeline control has reduced the number of arithmetic elements and implementation complexity. The basic idea to realize the two dimensional convolution circuit in a simple hardware is as follows. Eq. (3-1) is expanded in the columnwise way as

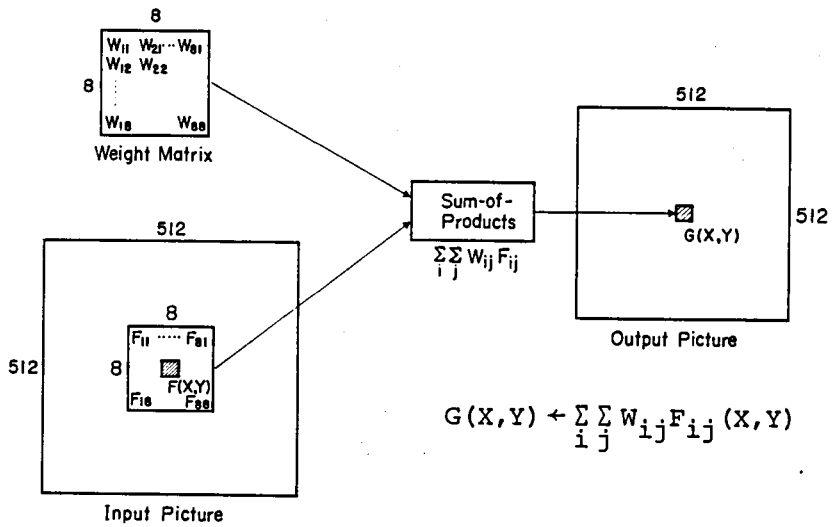


Figure 3-13 Two Dimensional Convolution Framework

$$G(x,y) = \sum_{j=1}^{m} W_{1j}F_{1j}(x,y) + \cdots\cdots + \sum_{j=1}^{m} W_{mj}F_{mj}(x,y). \qquad (3\text{-}18)$$

By the use of a newly designed circuit shown in Fig. 3-14, each term of Eq. (3-18), a partial sum-of-products, can be calculated with m multipliers and m-1 adders in parallel. The m terms of partial sums-of-products can be obtained successively by shifting both columnwise weight data and picture data, and it can be accumulated into the register of Fig. 3-15 to complete the calculation of G(x,y). This method, however, requires the high speed shifting capability for both weight matrix and line buffer in Fig. 3-14, which causes hardware cost to increase.

We can notice that a partial column of input picture data $F_{ij}(x,y)$, $j = 1, 2, \cdots m$, is used for all the calculation of $G(x,y)$, $G(x+1, y)$, $\cdots$, and $G(x+m, y)$ in common. If partial sums-of-products for these partial column data in $G(x,y)$, $\cdots$, and $G(x+m, y)$ are calculated, when the column picture data are shifted simultaneously at the line buffer memory, then the access time to fetch the picture data can be reduced to nil. The partial sums are

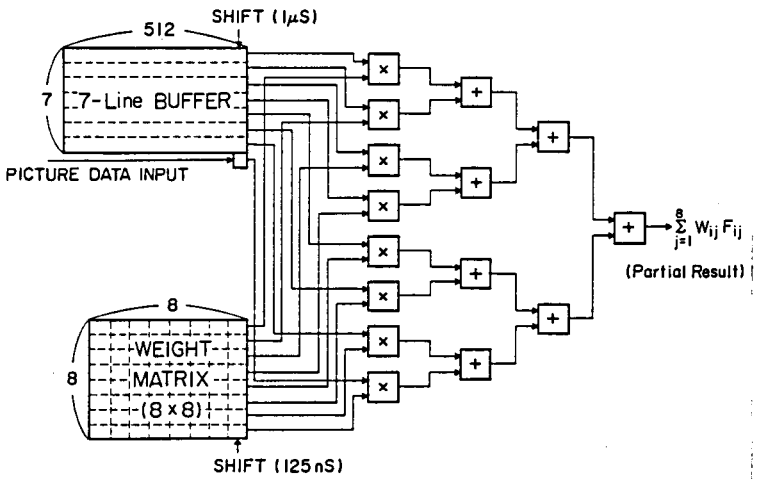


Figure 3-14 Two Dimensional Convolution Circuit Blockdiagram (Partial Sum-of-Products)

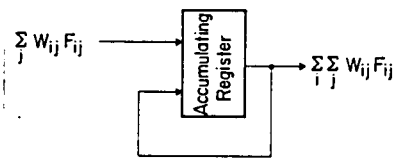


Figure 3-15 Accumulating Register for Completing Convolution

sequentially accumulated into 8 registers, which are shifted synchronizing with the line buffer, as shown in Fig. 3-16. The accumulation is done by the pipeline control to complete the convolution in such a way as flowing convolution results through the pipelined registers. As a result, by combining the circuit to calculate a partial sum-of-products with the pipeline accumulating control, the two dimensional convolution can be designed to include m multipliers and m adders instead of m^2 multipliers and m^2-1 adders.

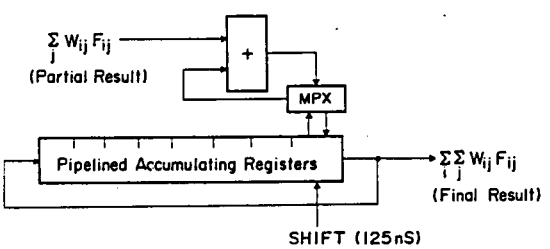


Figure 3-16 Pipeline Accumulating Control (Two Dimensional Convolution)

As a result of trade-offs among speed, complexity and economy, the size of local window was determined to be eight (m = 8). Every partial sum-of-products is calculated at the clock rate of 125 ns, and 1 μs is needed to complete the calculation of Eq. (3-1) with a certain time of pipeline delay. This local parallel operation executes 128 data accesses (64 for local window and 64 for weight matrix), 64 multiplications and 64 additions within 1 μs.

Parallel access in the local window (8 x 8) is accomplished with the aid of 7 lines of picture data buffer and time-shared pipeline control. Eight columnwise pixel data are transferred to the convolution circuit in parallel and pipeline control to compute a partial sum-of-products simulates 8 linewise weight data access at high speed.

The convolution module operates on either 8-bit unsigned positive data (0 ~ 255) or 7-bit signed integer data (-128 ~ 127), and yields 22-bit unsigned positive data or 21-bit signed integer data, respectively. The output value may as well be normalized to 8-bit data. The adder adds a prespecified bias B and the shifter gives rightward shift by a given number n. That is,

$$Z = \frac{V + B}{2^n} \tag{3-19}$$

where Z is the normalized value, V the output of the convolution module, B a bias, and n a shift number. This normalization is performed by the circuit shown in Fig. 3-17. If the normalized data exceeds an 8-bit word, the overflow status is signaled to the host computer.

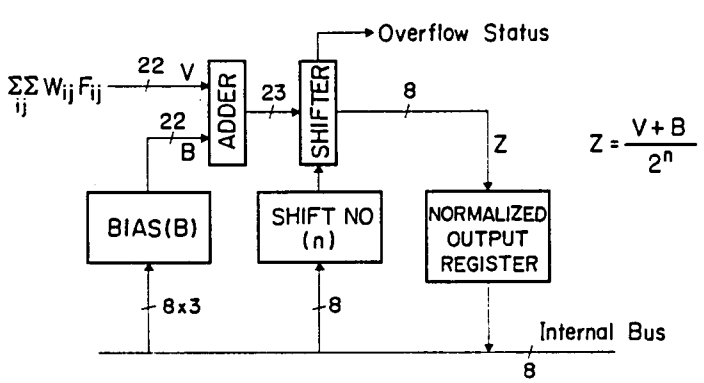


Figure 3-17 Normalization of Convolution Output

Any spatial filtering defined with the 8x8 weight matrix can be realized by this two dimensional convolution module. A set of weighting coefficients W_{ij} is transferred from the host computer piror to the execution.

(2) Logical Filtering

The logical filtering works on binary picture data for thinning, boundary detection, line segment direction coding, topological feature extraction, noise cleaning etc. Let us consider the following examples:

Thinning

There exist considerable literatures on various thinning schemes. One of the simplest algorithms is given by a set of 3x3 local masks, as shown in Fig. 3-18. The eight masks are applied in the sequence S1, S2, ..., and S8. A "1" of the center in a 3x3 window is deleted (i.e., is changed to "0") if its 3x3 neighborhood matches any mask.

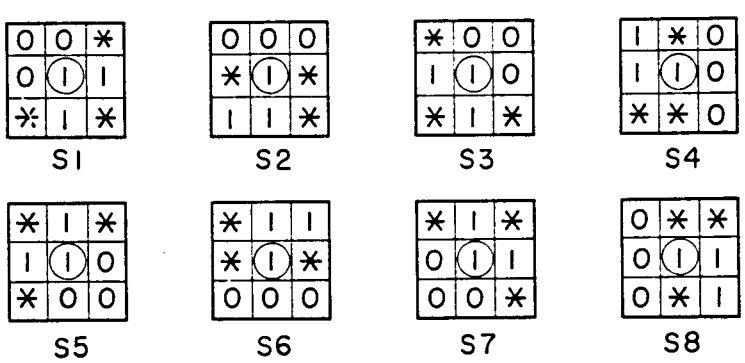
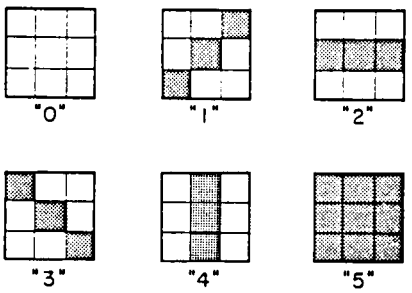


Figure 3-18 Thinning Masks

Boundary Detection

This operation detects the border between object and background. All the pixels whose value is "1" (object), and which have

at least one neighbor of "0" within their 3 x 3 local windows, remain as "1". They form the object boundary. All other pixels are set to "0".



(" " indicates the direction code.)

Figure 3-19 Direction Code of Line Segment

Line Segment Direction Coding

A set of 3 x 3 local masks can classify the direction of line segments into 6 categories, according to the bit configurations in the local window shown in Fig. 3-19. This coding is frequently used in character recognition.

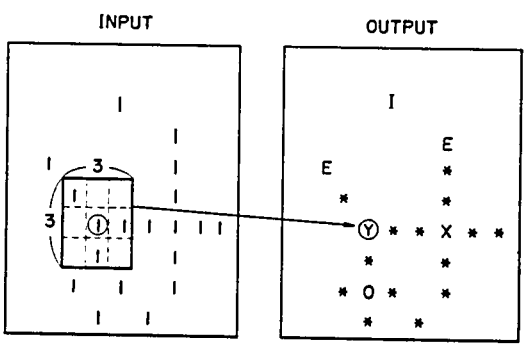


Figure 3-20 Topological Feature Extraction

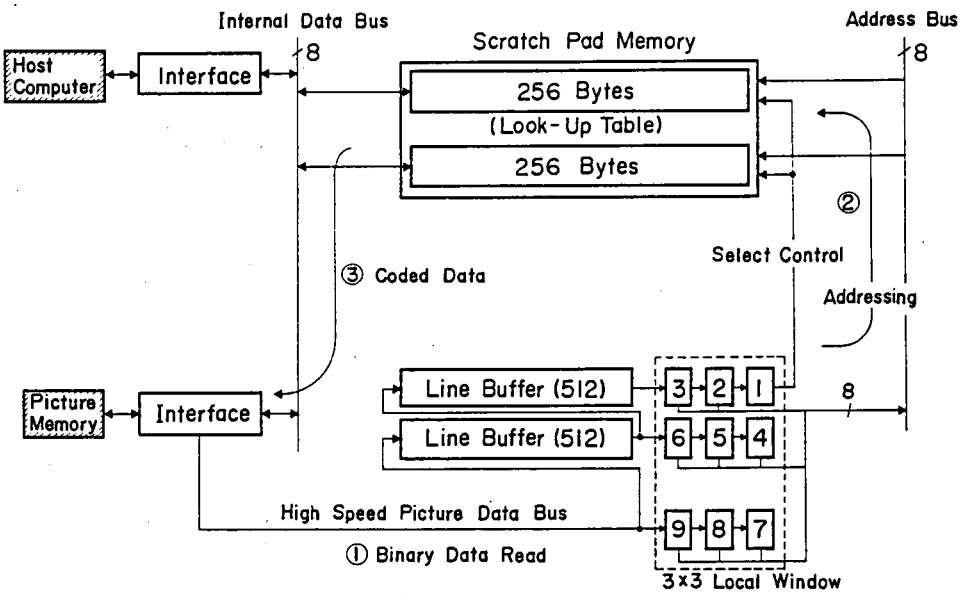
Topological Feature Extraction

This operation extracts from a thinned binary picture by a set of 3x3 local masks, end point (E), branching point (X-type, Y-type, or T-type), bending point (D), isolated point (I), and small loop point (0), as illustrated in Fig. 3-20. These extracted features are used in character recognition, bubble chamber negative analysis etc.

Noise Cleaning

A noisy binary picture can be filtered to output a smoothed one. A filtering operation employs a 3 x 3 local window to eliminate isolated small islands or speckles of black "1" and to fill in isolated gaps or notches in black "1" areas.

Most of these logical filtering operations can be implemented by the local parallel operation with a 3 x 3 window. Fig. 3-21



ⓘ indicates the i-th operational sequence.

Figure 3-21 Logical Filtering Module

illustrates the logical filtering module in PPP. The module takes every 3x3 local window in an input binary picture (①) and converts the 9-bit data to the output code by referring to the conversion table stored in the scratch pad memory; the 9-bit data in a 3x3 window represents a unique address of the scratch pad memory with a 512 byte capacity for a table look-up (② , ③).

The logical filtering module is equipped with the 2-line buffer memory and associated hardware to access a 3x3 window of picture data in parallel. The most significant bit in the 9-bit data controls the byte selection of the scratch pad memory, while other 8-bit defines an memory address. Thus, the scratch pad memory is considered as a dual 256-byte table.

Any logical operation defined on a 3x3 window is realizable by this module; an appropriate look-up table can be sent to the scratch pad memory from the host computer prior to the operation.

(2) Region Labeling

Region labeling is a local operation which labels the connected components of a given binary picture. The region labeling module constructs a picture where the "1" points are given a positive region number of label (greater than 1), two points having the same number if and only if they belong to the same connected component of "1" points in the original picture.

We need to define the "connectivity" in the picture. There are three types of pixel tesselation in digitized pictures: triangular, square and hexagonal arrays. PPP has chosen the square array because it is more practical in the sense of geometrical relations, conventional input device control, and so forth.

Two definitions of connectivity are possible in the square array as shown in Fig. 3-22. In the 4-connectivity mode, a point is called "connected" if there exists a horizontal or vertical neighbor, while in the 8-connectivity mode, a point is connected to any of its eight neighbors, including the diagonal ones.

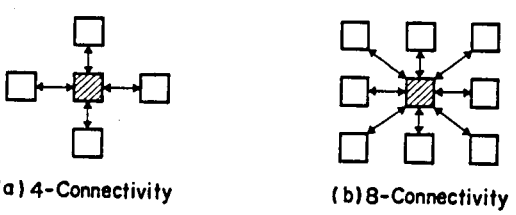


Figure 3-22 4-and 8-Connectivity

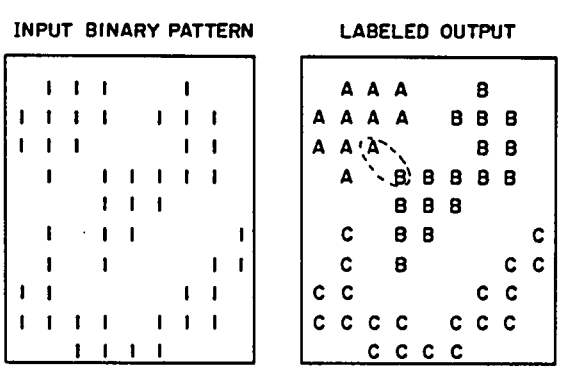
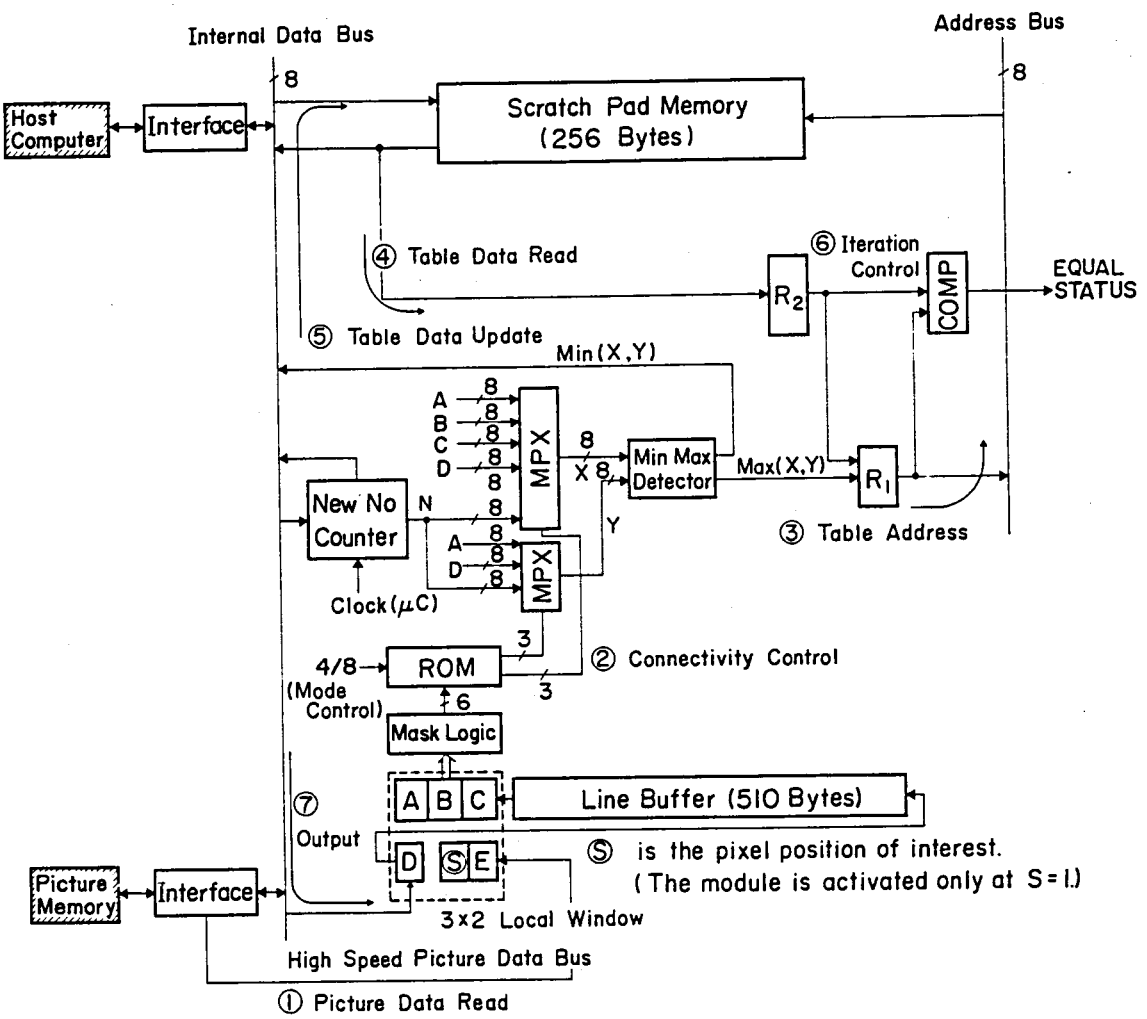


Figure 3-23 Region Labeling with 4-Connectivity

In the example of Fig. 3-23, the 4-connectivity mode is

applied. If the 8-connectivity mode is used, regions A and B would be decided to be connected and assigned by the same label.

Fig. 3-24 shows the region labeling module together with the scratch pad memory. The module includes a line buffer memory, read only memory (ROM), a new number (NN) counter, and associated circuits. A 510-byte line buffer memory is used to form a 3x2 local window for testing connectivity as well as to store the label number in one previous horizontal line. An 8-bit NN counter and two sets of ROMs control the data access of the scratch pad memory with associated multiplexers (MPX), min-max detector and two holding registers (R1 and R2). The region labeling module is activated only when the position S of a 3x2 window is "1" (①).



ⓘ indicates the i-th operational sequence.

Figure 3-24 Region Labeling Module

The scratch pad memory stores the connectivity relationship among labeled numbers under a microprogram control, while the NN counter determines a new label of connected components being sequentially detected by the raster scan. The 6-bit configuration with a mask logic, corresponding to a binary pattern in a 3x2 local window, are fed through the ROM to select one label of either previsouly labeled number (A, B, C or D) or new number (N) in Fig. 3-24. The selection is accomplished by two sets of MPXs. The connectivity mode selection is controlled with the aid of two sets of ROMs, whose contents are listed in Table 3-2 (②).

Comparing two selected label numbers, a greater one becomes the scartch pad memory address, while another smaller one is presented on the internal data bus (③). Then, the connectivity relation is updated in the scratch pad memory using the above address and data (④,⑤). This updating operation is repeated under a microprogram control until the address information (R1) is equal to the memory content (R2) by iteratively alternating the address with the content (⑥). At the end of updating operation, the microprogram controls the data transfer on the internal data bus to the position D in the local window and also to the picture memory via the interface (⑦).

The connectivity test operation is first performed in the raster scan mode, outputing the labeled numbers into the picture memory and storing the relation table in the scratch pad memory. After the completion of this operation, the microprogram edits the table data to eliminate the redundancy of connectivity relations in the scratch pad memory. Then, the currently labeled numbers will be mapped in final labels by the data conversion with the processed table data.

The labels used for processing are 8-bit numbers and PPP can accomplish the region labeling for at most 254 connected components. If there are more than 254 connected components, this processing will be repeated as many times as necessary for the

(S=1)					4-Connectivity			8-Connectivity		
A	B	C	D	E	X	Y	N.N.	X	Y	N.N.
0	0	0	0	0	Z	N	U	Z	N	U
0	0	0	0	1	Z	N	U	Z	N	U
0	0	0	1	0	Z	D	*	Z	D	*
0	0	0	1	1	Z	D	*	Z	D	*
0	0	1	0	0	Z	N	U	Z	C	*
0	0	1	0	1	Z	C	*	Z	C	*
0	0	1	1	0	Z	D	*	D	C	*
0	0	1	1	1	D	C	*	D	C	*
0	1	0	0	0	Z	B	*	Z	B	*
0	1	0	0	1	Z	B	*	Z	B	*
0	1	0	1	0	Z	B	*	Z	B	*
0	1	0	1	1	Z	B	*	Z	B	*
0	1	1	0	0	Z	C	*	Z	C	*
0	1	1	0	1	Z	C	*	Z	C	*
0	1	1	1	0	Z	C	*	Z	C	*
0	1	1	1	1	Z	C	*	Z	C	*
1	0	0	0	0	Z	N	U	Z	A	*
1	0	0	0	1	Z	N	U	Z	A	*
1	0	0	1	0	Z	D	*	Z	D	*
1	0	0	1	1	Z	D	*	Z	D	*
1	0	1	0	0	Z	N	U	A	C	*
1	0	1	0	1	Z	C	*	A	C	*
1	0	1	1	0	Z	D	*	D	C	*
1	0	1	1	1	D	C	*	D	C	*
1	1	0	0	0	Z	B	*	Z	B	*
1	1	0	0	1	Z	B	*	Z	B	*
1	1	0	1	0	Z	B	*	Z	B	*
1	1	0	1	1	Z	B	*	Z	B	*
1	1	1	0	0	Z	C	*	Z	C	*
1	1	1	0	1	Z	C	*	Z	C	*
1	1	1	1	0	Z	C	*	Z	C	*
1	1	1	1	1	Z	C	*	Z	C	*

("0" means all 0 in 8 bits
"1" means non-zero)

Z : Zero
U : Count up New No. Counter
* : None

(A, B, C, and N are labeled mumbers as shown in Fig. 3-24.)

Table 3-2 ROM Contents for Connectivity Test

remaining "1" data.

3-6 Supplementary Remarks

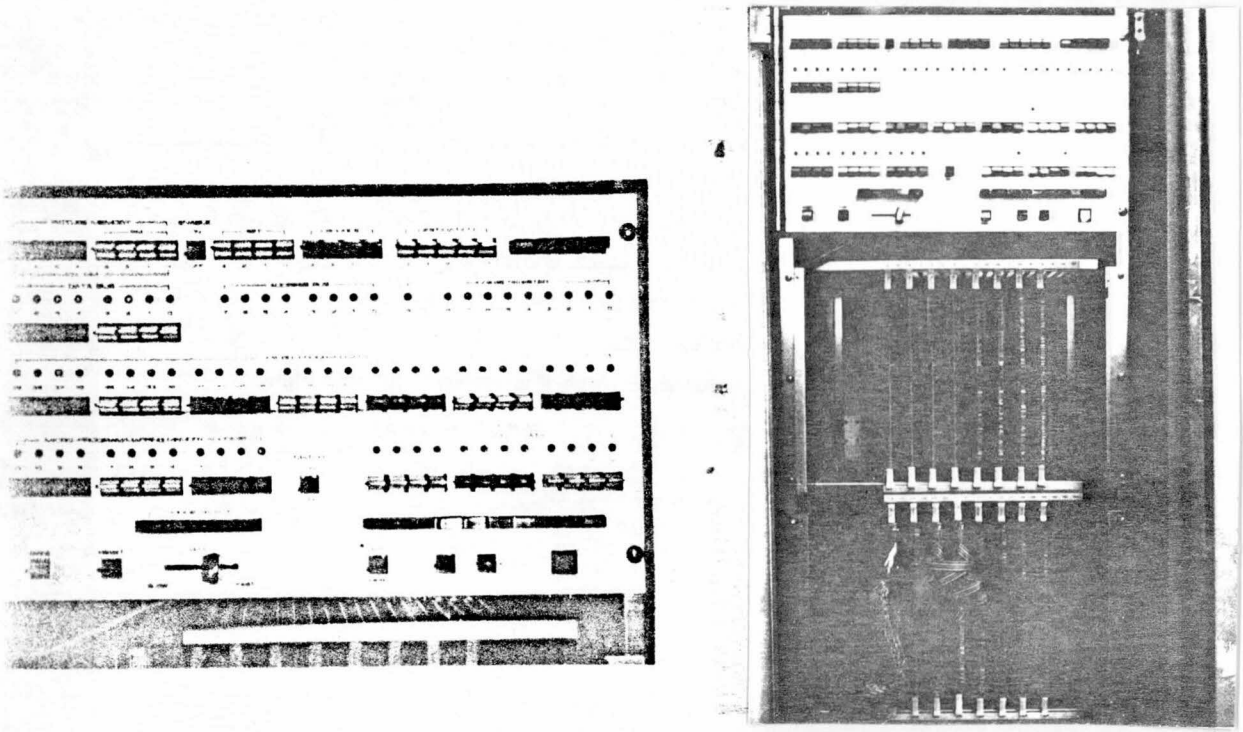
In this chapter, the hardware architecture of a newly developed local parallel picture processor (PPP) was presented in detail. The remarkable features in the design concepts as well as in the hardware realization were also given. Salient features of PPP and the picture memory are summarized in Table 3-3.

LOCAL PARALLEL PICTURE PROCESSOR
Local Parallel Operations — 8×8(Convolution), 3×3(Logical Filtering), 3×2(Region Labeling)
High Speed Picture Processing — 1μs/Pixel (Approximately 0.26 sec for 512×512 picture)
Microprogrammable Control — 40-Bit Microinstruction, 250 ns Clock Cycle
8-Bit Interal Data Bus and High Speed Picture Data Bus
Specialized Functional Modules — Convolver, Scratch Pad Memory, ALU/Multiplier
PICTURE MEMORY
Picture Size — 512 × 512
Picture Capacity — 4 Gray Scale Pictures and 4 Binary Graphic Planes
Data Transfer Rate — 1MHz Word (36-Bit Length)

Table 3-3 Salient Features of PPP and Picture Memory

Fig. 3-25 shows an overall view of an interactive picture processing system where PPP is included. The system consists of a high precision film scanner/recorder (left); a host computer

Figure 3-25 Overall View of Interactive Picture Processing System



(a) Console Panel

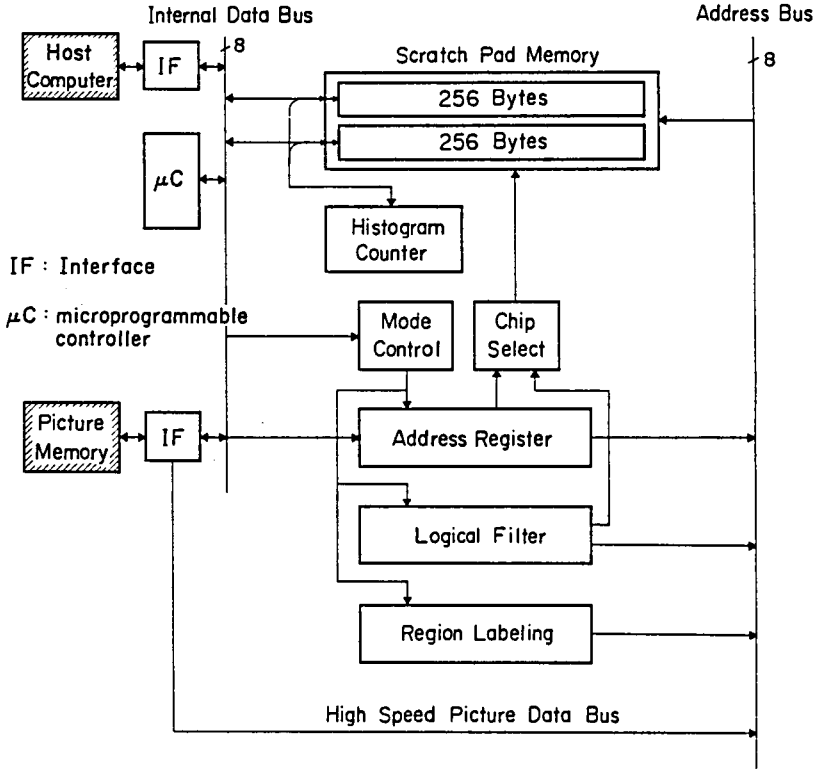
(b) Inside Assembly

Figure 3-26 View of Local Parallel Picture Processor

TOSBAC-40C (middle in the rear); PPP and the picture memory (right); a console CRT and a color monitor display (front); and a large capacity disk file storage unit (most right). The PPP console panel photograph is shown in Fig. 3-26 (a), while Figure 3-26 (b) indicates that the actual hardware assembly is as small as a conventional minicomputer. The following are the remarks regarding for the improvement and further expansion of PPP.

(1) Scratch Pad Memory Organization

The scratch pad memory is very efficiently used in the logical filter, region labeling, data conversion, and histogram computation. It consists of 512 bytes of high speed static RAM with 55 nS cycle time. Fig. 3-27 summarizes the configuration of the scratch pad memory and associated circuit.



Figture 3-27 Scratch Pad Memory Organization

Data access in the scratch pad memory is via the internal data bus in byte unit (8 bits) according to the content of the address bus. Each picture processing functional module can access the scratch pad memory through the address bus which is directly connected with the high speed picture data bus. On the other hand, the microprogram controller accesses the scratch pad memory through an address register between the internal data bus and address bus.

More bit length in the scratch pad memory can improve the function of histogram computation. As mentioned in Section 3-3, the word length (16-bit) of the memory limits the maximum counting number of 65,383 picture elements ($2^{16}-1$). For a larger picture size, more than an 18-bit word will be required to count the maximum pixel number (for example, in a 512 x 512 picture).

(2) Other Improvement Factors

PPP, at the current status, has proved its feasibility and applicability in the picture processing studies. However, it can be improved further by the following architecture changes:

(a) Local parallel operations such as two dimensional convolution need a set of line buffers to enable parallel data access in the local window. Shift registers, which are used as the line buffer memory at present, does not allow an arbitrary size of picture to be processed. When a user specifies a rectangular region of less than 512 width, the system program enforces the width to be 512 because of the local parallel operations in spite that. The random access memory and associated circuit permit an arbitrary sixe of regions.

(b) In executing the two dimensional convolution, the normalization factors (i.e., bias and shift number) should be predetermined by the user's program at present. The host computer has no way to optimize these normalization factors even with the overflow status acknowledgement. A min-max detector is expected to find

the minimum and maximum values as well as the positions at which they occur. These data can be used for the automatic normalization and correlation procedure.

(c) More advanced integrated circuit fabrication technology will reduce hardware complexity in terms of the number of IC chips and associated wirings, and also improve the execution speed. A one-packaged 8-bit by 8-bit multiplier, for instance, has become commercially available. In addition, several arithmetic and logical circuits have been integrated and higher processing speed is going to be realized.

(3) Further Expansions

The two dimensional Fourier transform and the maximum likelihood estimation will be the next functions for the hardware realization. At the hardware implementation of these functions, the bit length should be discussed in terms of the computation precision, hardware complexity, and execution speed. A floating point data will be required to avoid the computation overflow and/or underflow.

Another expansion will be a newly designed picture memory. A flexible address controller will be included in the picture memory, which can support versatile memory accesses for the graphic data generation, virtual array assignment, and so on. The number and size of pictures are strongly dependent upon the IC fabrication technology. If more integrated memory chips are commercially available, a new picture memory with the flexible address control will be practical; it can store more than 4 pictures of larger than 512 x 512 (perhaps 1,024 x 1,024 or 2,048 x 2,048).

102 項欠

CHAPTER 4 SOFTWARE ARCHITECTURE OF LOCAL PARALLEL PICTURE PROCESSOR (PPP)

The software architecture on both the host computer and the local parallel picture processor PPP is presented in this chapter.

Section 4-1 discusses an organization of the command program in the host computer. Software interface between the host computer and PPP is described, and a repertoire of basic function commands are listed. Section 4-2 gives detailed descriptions of the micro-program controller and instruction set. Section 4-3 discusses how to program PPP. A typical example of programming PPP will be listed for executing basic functions. Performance measurements will be given in comparison with those by the host computer (mini-computer) and a large scale general-purpose computer in Section 4-4.

Section 4-5 shows some applications of PPP. PPP operations have been performed on human face photographic data and remotely sensed data.

4-1 Functional Commands for PPP

The local parallel picture processor (PPP) has been specifically designed to perform the following picture processing functions at high speed: two dimensional convolution, affine coordinate transformation, logical filtering, region labeling, data conversion, histogram computation, and pixelwise operations.

These operations should be executed without any complex programming in the host computer. If it is easy to program the data transfer and function execution in the host computer, users can assemble efficient and flexible picture processing programs with minimum loss of time. PPP has employed the following procedures to facilitate the programming.

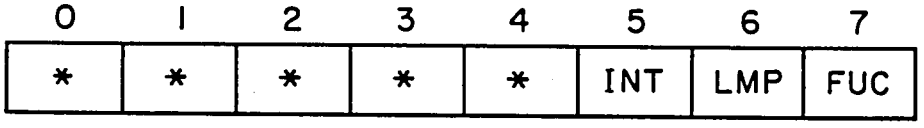
The PPP operations are initiated by simple command codes together with some parameters transmitted from the host computer. Then, they proceed independently by the microprogram controller until termination, leaving the host computer free to carry out other tasks in the interim. Upon completion, the microprogram signals an end-of-operation interrupt to the host computer.

The host computer is a TOSBAC-40C minicomputer with 64K bytes core memory. PPP is connected to the 8-bit (byte) I/O data channel of the computer. The following computer instructions control data flow paths on the I/O channel: output (write) data (WD), input (read) data (RD), output command (OC), and sense status (SS).

A WD computer instruction is used for transmitting microprograms, weighting coefficients, tabular data for the scratch pad memory to PPP, and an RD instruction for receiving histogram data. An OC instruction initiates PPP or specifies PPP operational modes, while an SS instruction is used for synchronizing the data transfer and acknowledging PPP status signals. Figs. 4-1 and 4-2 shows the command byte and status byte configurations, respectively.

An INT command initializes the internal status of PPP: status registers, flip-flops, and so forth. An LMP command is used for transmitting user's microprograms into the random access program memory. Primitive programs for basic picture processing functions can be stored in the read only program memory. An LMP command is also useful to debug microprograms. An FUC command starts PPP to execute a certain function specified by the command code

and argument words. A microprogram is defined as this code number in PPP and activated by an FUC command. At the end of a picture operation, an 8-bit status signal is returned to the host computer.



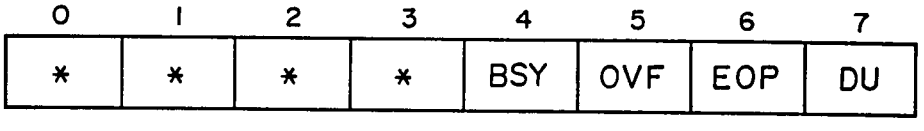
bit #

5 INT Initialization

6 LMP Load Microprogram

7 FUC Execute Picture Processing Functions

Figure 4-1 Command Byte Configuration



bit #

4 BSY Busy Status

5 OVF Overflow Error Status

6 EOP End-of-Operations

7 DU Device Unavailable

Figrue 4-2 Status Byte Configuration

An OVF error status signal is turned on when a data overflow occurs in histogram computation, output normalization of two dimensional convolution, new number counting of region

labeling, and ALU operations. A BSY (Busy) status signal is used to synchronize the data transfer between the host computer and PPP interface. An EOP (End of Operation) status signal is issued upon completion of picture processing operations. A DU (Device Unavailable) means PPP off-line mode or power-off status.

There are two types of operational functions issued with an FUC command: data transfer and picture operation. Data transfer commands are used for data communication between PPP and the host computer: histogram data, data conversion table, logical filter output table and weight matrix data. Picture operation commands are designed to execute the corresponding picture processing functions in PPP. Table 4-1 lists a set of functional commands with their code and arguments. Details are explained as follows.

Function	Code	Mnemonic	Arguments
Data Transfer	0	DTR	IW/BA/BL/BR
	1	DTW	SPM/WM, IW/BA/BL/BR
Picture Operation	2	FLT	OUT, IN, WND, MODEF, BIAS, SHIFT, STATUS$^{\Delta}$
	3	LFL	OUT, IN, WND, STATUS$^{\Delta}$
	4	LAB	OUT, IN, WND, MODEL, STATUS$^{\Delta}$
	5	AFN	OUT, IN, PAR, MODEA, STATUS$^{\Delta}$
	6	DCV	OUT, IN, WND, STATUS$^{\Delta}$
	7	HST	IN, INWND, MODEH, STATUS$^{\Delta}$
	8	EXM	OUT, IN, WND, MADRS, STATUS$^{\Delta}$

Table 4-1 Functional Commands for PPP
(Δ indicates a return-status word from PPP.)

(1) Data Transfer Command

Data Read from Scratch Pad Memory (DTR)

A DTR command allows data transfer from PPP to the computer according to the data control argument. For instance, the data of computed histogram is transmitted by a DTR command with 16-bit integer word mode (IW).

Data Write in Scratch Pad Memory or Weight Matrix Memory (DTW)

64 weighting coefficients used for the two dimensional convolution are stored into a weight matrix memory (WM) by a DTW command. A DTW command also transmits 256-byte conversion table data for data conversion or 512-byte output code data for logical filtering.

The scratch pad memory (SPM) can be set to one of four different data access modes under program control: 256 sixteen-bit integer words (IW), 512 bytes (BA), 256 bytes-left half only (BL), and 256 bytes-right half only (BR).

(2) Picture Processing Function Execution Command

PPP carries out each basic picture processing function under program controls. In general, a picture operation requires a pair of operands for specifying input and output pictures, address parameters for scanning the pictures and checking the boundary, a set of parameters for selecting operation modes etc. When the operation is completed, the status information is returned to the computer via the interface.

A typical sequence of commands would have the following steps:

(i) Transfer tabular data to PPP by a DTW command.
(for Two Dimensional Convolution, Logical Filtering, Data Conversion)

(ii) Issue an execution command of picture operation as follows:

Assign the picture memory to output and input (OUT and IN).

Set the parameters in the address controller (WND or PAR).

Set the parameters or select the mode (MODE*).

Sense the status information (STATUS).

(iii) Read a computed data from PPP by a DTR command.

(for Histogram Computation)

The parameters for memory assignments (OUT and IN) have special bit configuration as shown in Fig. 3-4. The host computer must specify the same input and output assignment to the picture memory. The block of address parameters (WND) contains a set of address values stored in registers of Figs. 3-6 and 3-8. Each parametric value is computed in the host computer, according to a given boundary information, the size of local mask, and pipeline delay. Most picture processing operations, except the affine coordinate transformation (AFN), scan the picture with normal raster. An AFN command requires all the parametric values (PAR) in Eq. (3-11) which specify a given coordinate transformation. An HST command uses only input parameters.

A status information (STATUS) is returned to the host computer, which indicates a termination condition of PPP (e.g., normal end, overflow error etc.) upon completion of picture operations.

The following picture operation commands are issued from the host computer with their parameters.

Two Dimensional Convolution or Spatial Filtering (FLT)

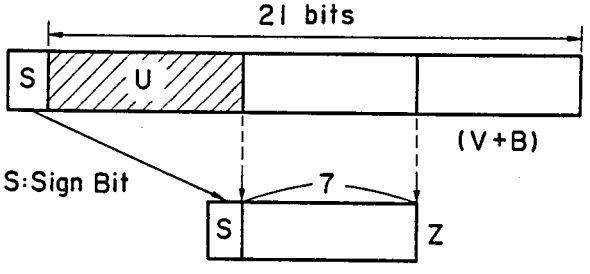
An FLT command accomplishes the two dimensional convolution of a picture with an 8x8 weight matrix. The mode selection controls the data type of picture and weights information: unsigned 8-bit positive integer (0 ~ 255) or signed 8-bit integer (-128 ~ 127). The unsigned data mode is used for averaging, correlation etc.,

and the signed data for derivative operations, enhancement, template matching etc.

The computation results in 21-bit or 22-bit integer data for unsigned mode or signed mode, respectively. Usually, the resultant output is normalized to 8-bit integers for a convenience to storing and to further processing. The normalization is specified by two arguments, bias B and shift number n as follows:

$$Z = \frac{V + B}{2^n}$$

where Z is the normalized value and V is the convolution output. The normalization allows any contiguous 8 bits of the biased value to be shifted as shown in Fig. 4-3.



If S=0 and 1 in U part : Overflow Status

If S=1 and 0 in U part : Overflow Status (Underflow)

(V : Signed Integer)

Figure 4-3 Normalization and Error Status

The overflow error status is returned to the host computer if S = 0 (positive) and there is any "1" in U part, or if S = 1 (negative) and any "0" in U part.

A matrix of weight data is transmitted by a DTW command prior to an FLT command.

Logical Filtering (LFL)

An LFL command operates on a binary picture which is specified by an input select word (IN); it is one of the four binary picture data stored in the plane memory. A binary picture, for example, can be produced by thresholding a gray scale picture with the data conversion function.

9-bit data in a local 3x3 mask represents a unique scratch pad memory address and an 8-bit output code of the reference address is looked up from the memory. The scratch pad memory should be loaded from the host computer with an appropriate table of transformation by a DTW command. The programmable scratch pad memory permits PPP to accomplish any kind of logical filters.

Region Labeling (LAB)

An LAB command is also applied to the binary picture data, which is fed through a selected plane memory. The region labeling module tests the connectivity in each 3x2 local mask, incorporating the scartch pad memory as a buffer register which stores the connectivity relationship among labeled numbers. The connectivity mode is defined by the command argument: either the 4- or 8-connectivity.

Due to the byte size of the picture memory (i.e., 8 bits), the maximum number of labels is 254 (from 2 to 255; 0-background, 1-object region in the original picture).

Linear Coordinate Transformation or Affine Transformation (AFN)

A linear coordinate transformation is defined by an AFN command and its address parameters (PAR). A set of address parameters should be priorly computed in the host computer. Since the calculated address value is not always an integer, the digital picture coordinate transformation needs resampling or gray scale interpolation. Three resampling algorithms can be built in PPP: nearest neighbor, linear interpolation, and cubic convolution.

One of three resampling algorithms is specified in an AFN command as an argument. The address calculator has an additional register to calculate the addresses of neighboring picture points according to a given resampling mode. Four neighboring points data and fractional address information are necessary for linear interpolation; the ALU and multiplier in the microprogram controller are used. Cubic convolution can be achieved by 16 neighboring points data and their address information. The ALU, multiplier, and scratch pad memory are used to reconstruct the amplitude at the resampling point.

Point Mapping or Data Conversion (DCV)

A DCV command executes the point mapping function which converts picture data pixel by pixel according to the content of scratch pad memory. The scratch pad memory plays a role of a high speed data transformation operator and is loaded from the host computer with an appropriate table.

Prior to a DCV command, a DTW command is used to transmit a set of transformation values to the scratch pad memory.

A DCV command with an identity transformation (i.e., $\phi(z) = z$) enables a picture data in one picture memory to move into another; this is frequently used for data interchange among the picture memory.

Histogram Computation (HST)

An HST command counts the frequency of gray levels within a given window area of an input picture. 256 registers are defined in the scratch pad memory. Each register is 16-bit length and is used for counting the number of pixels having the same gray level.

A mode control bit enables the histogram register to be initialized. If the register is not cleared, an HST command can accumulates the histogram information. When the occurrence of one gray level exceeds the 16-bit number, the overflow status information is returned to the host computer, and the count number

is set to 65,383 (2^{16}-1). This command can be also used for measuring the object area in a binary picture. After the completion of an HST command, the histogram information in the scratch pad memory is transmitted to the host computer by a DTR command.

Execute Microprogram (EXM)

An EXM command is used for executing any user defined microprogram. Pixelwise arithmetic and logic operations can be programmed with the aid of an ALU and a multiplier in the microprogram controller.

In addition to the pixelwise operations, more complex picture processings can be built in a combination of primitive functions. Several command sequences can be arranged into a macrocommand, which are designed of expedite a sequence of picture processing without any intervention to the host computer.

User's microprograms are loaded into the random access program memory by an LMP command, and then activated by an EXM command.

4-2 Microprograms in PPP

As described before, PPP is under microprogram control. The microprogram controller plays the following functional roles: communication to the host computer, interface to the picture memory, data flow control on the internal bus, control of functional modules, and pixelwise arithmetic and logical operations. The word length of a microprogram instruction is 40 bits and the cycle time is 250 nS. The microprogram instructions are designed to accomplish the above functions in one cycle as simultaneously as possible, i.e. in parallel. Picture operations are programmed with the microprogram instructions and then carried out by the

microprogram controller.

Fig. 4-4 shows the blockdiagram of the microprogram controller.

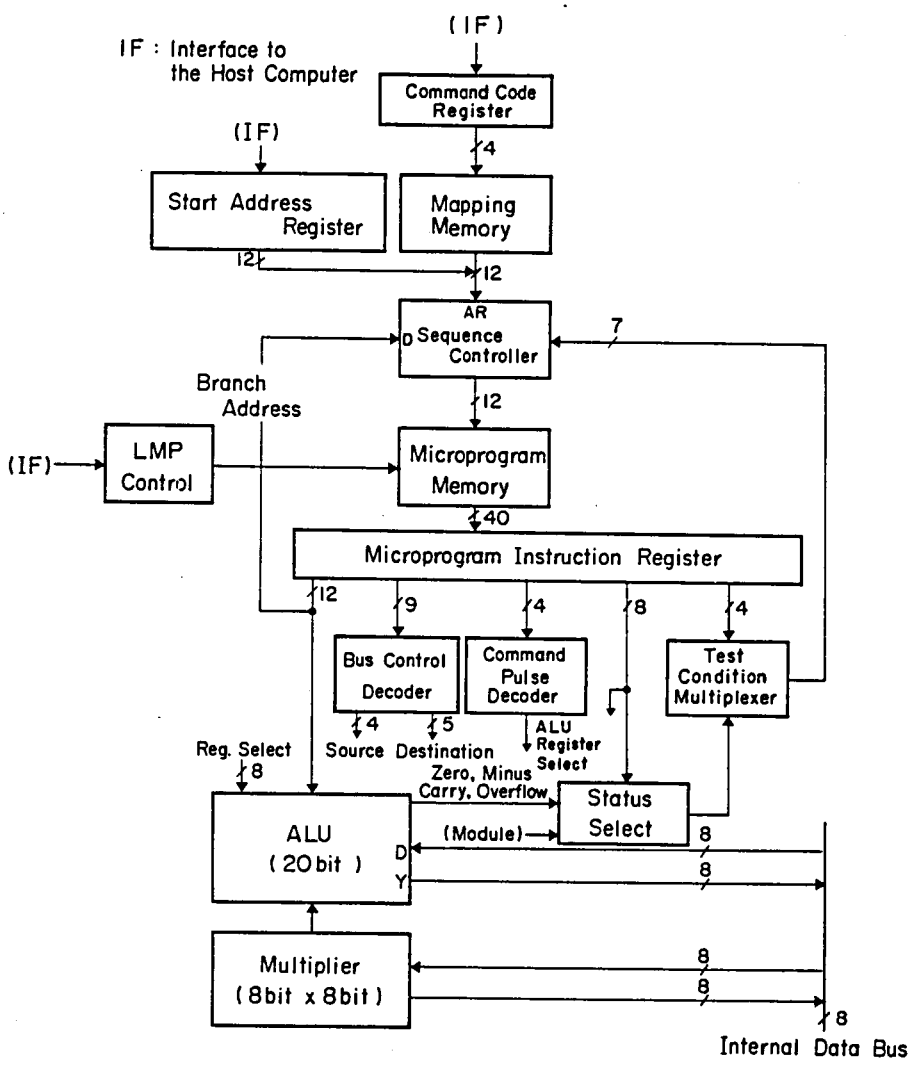


Figure 4-4 Blockdiagram of Microprogram Controller

The command code register receives a command code from the host computer, which is listed in Table 4-1. The command code determines the address of the mapping memory, whose output is used as the starting address of a given functional microprogram sequence. The starting address is then loaded into the micro-

program sequence controller, while an EXM command specifies the starting address of a user's microprogram. The microprogram sequence controller has two basic functions: it synchronizes external events with the microprogram control, and it uses the output of the test condition multiplexer to determine whether or not microprogram branches, jumps-to-subroutine, and returns-from-subroutine are to be made.

The microprogram memory, which can contain 4,096 (2^{12}) instruction words, are connected to the microprogram instruction register. The 40-bit microprogram instruction word, as shown in Fig. 4-5, specifies the ALU operation, data bus source and destination, command pulse generation, status selection and microprogram sequence. Details can be found in the following.

	Bit No. / Instruction	39 38 37 36 Sequence Control	35	34	33 32 31 30	29 28 27 26	25 24 23 22	21 20 19 18	17	16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
1	TRN	0 0 0 0	FLT	*	Command Pulse	Data Flow Control: Destination	Source	*						
2	TB	0 0 0 1	FLT	*	Command Pulse	Data Flow Control		*	True/False		Status Selection	Branch Address		
	CALL	0 0 1 0												
	RET	0 0 1 1												
	JMP	0 1 0 0												
3	ALUR	1 0 0 0	FLT	*	Command Pulse	Data Flow Control		A-reg			B-reg	ALU Operation: Function	Source	Destination	Shift
	ALUX	1 0 0 1	No of Iterations											
	ALUI	1 0 1 1	Immediate Data (20 bits)											
4	SAR	1 0 1 0	FLT	Bus/Map	Command Pulse	Data Flow Control		A-reg			B-reg	ALU Operation		

(* indicates the don't care bit field.)

Figure 4-5 Microprogram Instruction Word

The ALU operates on 20-bit data with several arithmetic and logic functions. Sixteen registers are contained in the ALU to facilitate the high speed computation. Ten registers out of 16

registers in the ALU are exclusively used for the input address calculators (as shown in Fig. 3-8).

The data exchange is provided with the internal data bus. As depicted in Fig. 4-6, the internal data bus carries 8-bit data from one functional module to another, which is under microprogram control as the source and destination via the bus. The command pulse generator synchronizes the PPP operation with the data bus control, and is responsible for microprogram execution.

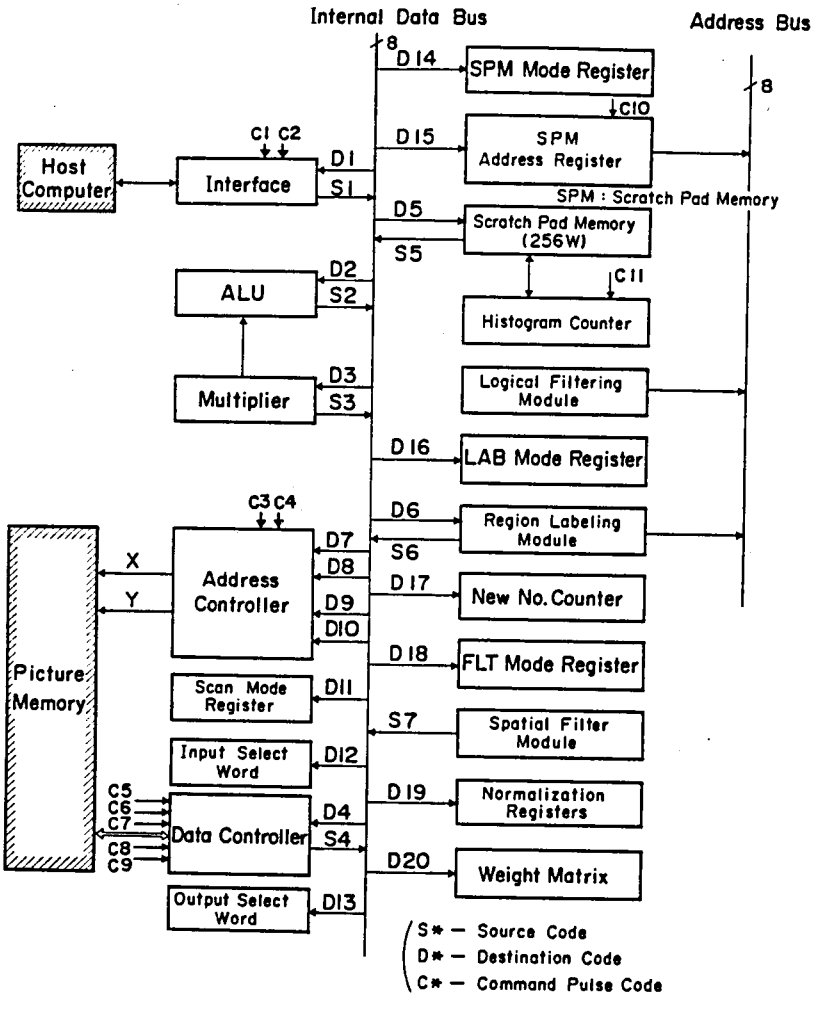


Figure 4-6 Data Flow Control (Source and Destination Codes)

There are four major types of the microprogram instructions which are defined by 4-bit sequence control: data flow control (TRN),

program branch control (TB, CALL, RET and JMP), microprogram address set (SAR), and ALU operations (ALUR, ALUX and ALUI). The command pulse generation, data flow control, branch control, and/or ALU operations might be combined into one instruction word to perform as many tasks as possible at once.

(1) Data Flow Control Instruction (TRN)

This instruction controls the data transfer via the internal data bus and also issues the command pulse. Four bits are provided for the data source determination and five bits for the bus destination selection. For instance, the picture data from the picture memory is fed on the bus as a source (S4) and supplied to the input of the ALU as a destination (D2); the computed data by the ALU is presented to the data bus as a source (S2) and transmitted to the picture memory as a destination (D4). (See Fig. 4-6.) Data transfer among the computer interface, scratch pad memory, special functional modules and mode registers are also under microprogram control.

Four-bit command pulses are used to synchronize the data transfer among the host computer, picture memory and PPP, and to control the timing of each processing function.

One control bit (FLT) in the instruction word controls the two dimensional convolution timing sequence in calculating the partial sums-of-products.

(2) Branch Control Instructions (TB, CALL, RET, JMP)

Four instructions of this type are involved in PPP: test and branch (TB), call subroutine (CALL), return from subroutine (RET), and jump on the sequencer address (JMP).

The sequence controller can select one microprogram address information out of the following four address data sources: external direct input address defined by the 12-bit branch address field

(TB and CALL), stacked address for nested subroutine linkages (RET), address data in the sequencer address register (JMP), and microprogram counter address (other instructions). The microprogram address can be set to all zeros, which allows a jump to the initial state.

The microprogram sequence controller operates in conjunction with the PPP internal status selection and condition test. Five-bit field determines the true or false test condition and the status selection. The PPP internal status signals include the status flags from the ALU (carry, overflow, zero, and minus), the scan-line-end and frame-end status from the address controller, status signals indicating the data transfer timing among the host computer, picture memory and PPP, and so forth.

The branch control instructions are also designed to issue the command pulse and control the data flow from a source to a destination via the internal data bus as simultaneously as they can.

(3) ALU Operation Instructions (ALUR, ALUX, ALUI)

A high speed microprocessor is included in the microprogram controller, which operates on 20-bit data. A microprocessor consists of 16 registers, a high speed arithmetic and logic unit (ALU), an accumulator, and the associated shifting, decoding and multiplexing circuits. The 12-bit ALU operation field is organized into 4 groups of three bits each, and selects the ALU function, source operands, destination register, and their shift control as shown in Fig. 4-5.

The high speed ALU can perform three binary arithmetic and five logic operations on the two input operands. One input operand is driven from a multiplexer to select one of the A-register, direct input, and "zero". Another input operand is one of the A-register, B-register, accumulator, and "zero". Two registers in any of 16 registers can be accessed simultaneously according to the 4-bit

A-register and B-register selection fields in the instruction word. A "zero" data is given by the multiplexer inhibition control. One of eight ALU functions is selected by the 3-bit field.

The ALU data output is routed to several destinations. It can be output directly or stored in the register or accumulator with shifting, which are defined by the 3-bit destination field as well as by the 3-bit external shift control field. Four status signals are designed to indicate carry, overflow, minus and zero conditions of the ALU output.

There are three types of ALU operation instructions. ALU operations are also designed to issue the command pulse and control the data flow on the internal bus as simultaneously as they can.

ALU Operations with Registers (ALUR)

ALU operations are performed on 20-bit data in A and B registers specified by the 4-bit selection fields. The same register can be specified, in which case the identical data will appear.

ALU Operations with Iteration (ALUX)

The 8-bit iteration counter enhances the programming capability of iterative execution of the same instruction. An ALU operation instruction can be executed any number of times (up to 255 times) by controlling the microprogram sequencer in conjunction with the iteration counter. The number of iterations is loaded into the counter as defined by the 8-bit field and then counted down each time the instruction is executed. This iterative ALU operation is used for continuous shifting, division, and so forth.

ALU Operations with Immediate Data (ALUI)

The 20-bit immediate data in the instruction word can be applied to one of the ALU source operands, the direct input. Another input of the ALU source operands can be assigned by the 4-bit B-register field to select one of 16 registers.

This immediate ALU operation is frequently used for handling a constant value in the microprogram. Especially, the input

address calculator needs the 20-bit constant values; for example, 0.0, 0.5, 1.0, and -1.0 are loaded into registers DXY, DXX, DX etc.

(4) Set Address Register Instruction

A special instruction (SAR) is designed to set a microprogram address into the sequence register. One bit (Bus/Map) field controls the address data source: the address data from the data bus or from the mapping memory. The initial microprogram address for the basic processing functions are stored in the read only mapping memory, which is activated by the command code. A user can load any microprogram into the random access program memory and then initiate it by specifying its start address.

An SAR instruction is also designed to perform the ALU operation, command issue, and data flow control simultaneously.

4-3 Programming PPP

As described in Sections 4-1 and 4-2, there are two design steps for programming PPP: programs in the host computer and in PPP itself.

A microprogram assembler, optimized specifically to support microprogram instructions listed in Section 4-2, has been developed to allow each instruction word to be coded with a combination of numerical designators and alphabetical mnemonics in a format. The assembled microprogram object codes are loaded into PPP microprogram memory with the aid of a special-purpose loader in the host computer. This loader also allows the linkage of required subroutines. An LMP command is used to transmit microprograms

from the host computer to PPP, and then executed by an EXM command to test the performance. Once the microprogram for each processing module is verified, it might be permanently stored into the read only memory (ROM). An LMP command together with the linkage loader, however, encouranges the frequent generation of microprograms designed to carry out complex processing operations with a combination of primitive functions or pixelwise operations with the ALU and multiplier.

From the programming point of view in the host computer, there are the following three types of command sequence to accomplish basic picture processing functions. (It is assumed here that a set of microprograms for each basic function is already loaded into the program memory, and picture data is also stored in the picture memory.)

(1) Picture Operation Commands with DTW Command

A set of tabular data is required in the weight matrix memory or scratch pad memory for the two dimensional convolution (FLT), logical filtering (LFL), and data conversion (DCV).

For instance, the two dimensional convolution program has the following steps in the host computer. Supposed that an input picture is stored in the picture memory M1 and that a resultant picture is expected to be output into the picture memory M2. (See Fig. 4-7.)

Step 1 Assign the picture memory M1 and M2 to input and output, respectively. This assignment is done by another interface to the picture memory.

Step 2 Transfer a weight matrix table to PPP by a DTW command. The weight matrix table is set interactively by a user.

Step 3 Issue the two dimensional convolution command FLT to PPP as follows:

(i) Set the input and output memory select word (OUT, IN).

(ii) Set the address parameters (WND).

(iii) Set the input data type mode (MODEF).

(iv) Set the normalization factor (BIAS, SHIFT).

(v) Start the execution.

(PPP works with the picture memory independently from the host computer.)

Step 4 Receive the interruption of PPP operation end and read the status word (STATUS).

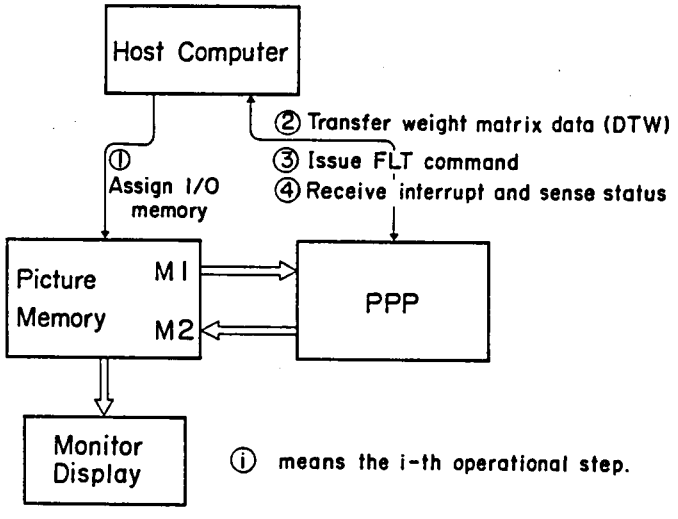


Figure 4-7 Programming Step of Two Dimensional Convolution

Other commands of the same type (LFL and DCV) are similarly carried out. The logical filtering command LFL requires to transfer a 512-byte output code data to the scratch pad memory from the host computer. The data conversion command DCV needs a 256-byte conversion data in the scratch pad memory before the command execution.

(2) Picture Operation Command with DTR Command

The histogram computation (HST) produces a set of table data (histogram data) which is transferred to the host computer by a DTR command. Fig. 4-8 shows the command sequence of the histogram computation.

Step 1 Assign the picture memory M1 to input.

Step 2 Issue the histogram computation command HST as follows:

(i) Set the input memory select word (IN).

(ii) Set the address parameter of rectangle window (INWND).

(iii) Set the histogram buffer mode (MODEH).

(iv) Start the execution.

Step 3 Receive the interruption and read the status (STATUS).

Step 4 Read the computed histogram data in the scratch pad memory by a DTR command.

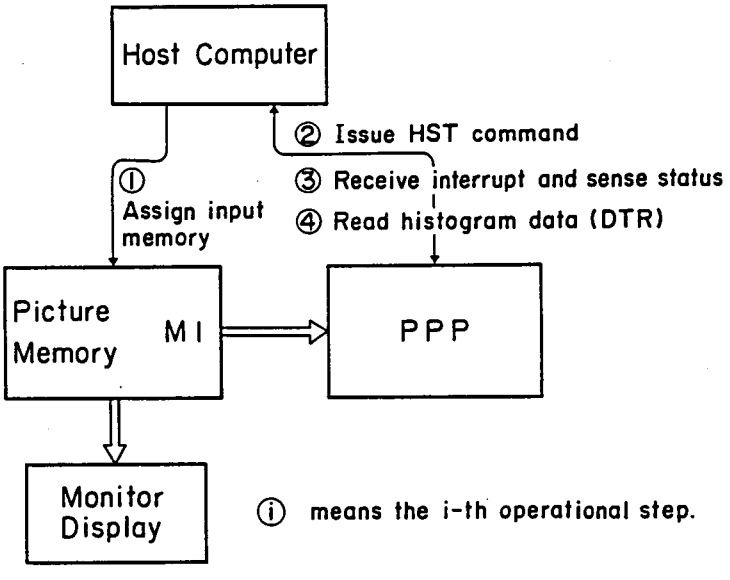


Figure 4-8 Programming Step of Histogram Computation

(3) Picture Operation Commands Only

The linear coordinate transformation (AFN) and region labeling (LAB) need no data transfer between PPP and host computer prior to command execution. An AFN command, for example, is performed as follows:

Step 1 Assign the picture memory M1 and M2 to input and output.

Step 2 Issue an AFN command as follows:

(i) Set the input and output memory select word (OUT, IN).

(ii) Set the address parameter (PAR).

(iii) Set the interpolation mode (MODEA).

(iv) Start the execution.

Step 3 Receive the interruption and read the status (STATUS).

The region labeling command LAB is similarly accomplished as in an AFN command.

It is one of the important feature of PPP that the microprograms tailored by user's own purposes can be loaded and executed. As mentioned before, any microprogram can be loaded into the program memory (RAM part) by an LMP command and then activated by an EXM command. A picture processing task of this type will be progressed according to the following sequence.

Step 1 Assign the picture memory to input and output.

Step 2 Transmit a package of microprogram instructions by an LMP command with the start address in the program memory.

Step 3 Issue an EXM command with the corresponding address of microprogram as follows:

(i) Set the input and output memory select word (OUT, IN).

(ii) Set the address parameter (WND).

(iii) Set the start address of microprogram (MADRS).

(iv) Start the execution.

Step 4 Receive the interruption and read status (STATUS).

The gradient operation on picture data, which calculates the magnitude and direction of gradient vector at each pixel, is one of the applications to show an assortment of basic picture processing functions included in PPP.

If the input is denoted by $F(x,y)$, then the magnitude $G(x,y)$ and direction code $D(x,y)$ of the gradient vector at point (x,y) can be obtained as,

$$\begin{cases} G(x,y) = \sqrt{\{\frac{\partial F(x,y)}{\partial x}\}^2 + \{\frac{\partial F(x,y)}{\partial y}\}^2} \\ D(x,y) = \Theta\,[\tan^{-1}\{\frac{\partial F(x,y)}{\partial y} / \frac{\partial F(x,y)}{\partial x}\}] \end{cases}$$

where $\partial F(x,y)/\partial x$ and $\partial F(x,y)/\partial y$ are output values of the two dimensional convolution with the x-derivative and y-derivative masks, respectively. Θ indicates a multilevel thresholding function which encodes an angle θ into one of 16 direction codes.

Let the input picture be stored in one of the picture memory denoted by M1. It is assumed that the gradient operation program is able to use other three picture memory (M2, M3, and M4). The main stream of program to calculate $G(x,y)$ and $D(x,y)$ is as follows. The output value consists of 4-bit magnitude $G(x,y)$ and 4-bit direction $D(x,y)$; they are packed into an 8-bit pixel data of the picture memory.

Main step 1 Execute the two dimensional convolution FLT with the x-derivative mask. The result is stored in M2.

This main step is performed step-by-step as discussed in an FLT command.

Main step 2 Similarily, execute the y-derivative operation for M1 picture data and store the result in M3.

Main step 3 Accomplish a special-purpose pixelwise operation which calculates the magnitude G(x, y) and then store the result in M4 as follows:

(i) Assign the picture memory M2 and M3 to input, and M4 to output.

(ii) Transfer the square-root conversion table to the scratch pad memory (DTW).

(iii) Transmit a microprogram to the program memory (LMP) and execute it (EXM).

A special-purpose microprogram calculates the squares of M2 and M3, separately with the aid of a multiplier included in the microprogram controller, and then add these two values in the ALU. Upper 8-bit data from the ALU is fed to the data conversion function to obtain its square root. The result, 4-bit magnitude G(x,y), is stored in M4.

Main step 4 Execute a pixelwise operation which calculates the division of M3 by M2, and store the result in M1. (The original picture is destroyed here).

(i) Assign the picture memory M2 and M3 to input, and M1 to output.

(ii) Transfer the inversion table to the scratch pad memory (DTW).

(iii) Transmit a microprogram (LMP) and execute it (EXM).

A microprogram looks up the inversion table by M2 picture data as an address to obtain 1/M2, computes M3*(1/M2) with the multiplier, and then saves the result in M1. The result is 8-bit data, and consists of 2 sign-bits of M2 and M3, and 6-bit magnitude of M3/M2.

Main step 5 Execute the data conversion function with a special conversion table θ ($\tan^{-1}x$) for M1 and store the result (the direction code D(x,y)) in M2.

The direction code is compromised of 4-bit data.

Main step 6 Accomplish a pixelwise operation for M4 and M2 data, and then store the final output in M3.

A pixelwise operation packs 4-bit magnitude and 4-bit direction data into an 8-bit data to be output.

4-4 Performance Measurements of PPP

PPP performance in terms of execution time for basic picture processing functions obviously depends on the speed and versatility of the picture memory, the host computer, and its software system. The PPP performance measurement which will be presented in this section has been done when the following hostcomputer and picture memory were used.

Host computer

The host computer is a minicomputer TOSBAC-40 model C with 64 KB core memory, whose cycle time is 0.8 microsecond. The basic instruction timing for register-operation is 1.0 μS (add), 6.8 μS (Multiply), and 13.3 μS (Divide). T-40C has two I/O channels: the multiplexer channel for slow speed I/O devices (max. 64 KB/sec) and the selector channel for high speed peripherals (max. 1 MB/sec). PPP is plugged in the multiplexer channel because of the ease in development of supporting software, while the picture memory is connected to the selector channel for high speed data transfer.

Picture Memory

The picture memory contains four 512x512 gray scale pictures with 8-bit pixel data, and four 512x512 binary pictures, a total of 36 bits per pixel. It consists of 4 K-bit dynamic RAM IC chips and associated circuits for multi-purposes. It is designed

as a high speed buffer memory for random access devices, such as PPP, picture I/O and host computer, and as a refresh memory for color display. Access time for external devices is 1 μS per pixel, while the refresh cycle is unavoidable due to the dynamic RAM; it causes the essential access time to slow down approximately 5%. Displaying the picture data is valid only at the time when it is not occupied by external devices. The video output electronics enables the composition of color signal (red, green and blue) of any picture memory banks.

PPP microprogram controller executes one microprogram instruction in 0.25 μS (at the speed of 4M instructions per second). Since PPP is connected to the host computer via the multiplexer I/O channel, the data transfer time between them is approximately 16μS per byte (at 64 KB/sec).

As described in Section 4-3, the basic picture processing functions require the following steps, with their execution time:

(i) Memory assignment to the picture memory ($\sim$ 10μS)

(ii) Data transfer to the weight matrix memory or scratch pad memory of PPP 1 mS $\sim$ 10 mS

(iii) Command issue of picture operations with a set of parameters
parameter set (memory selection, address, set mode control etc.) ($\sim$ 1 mS)
execution of picture operations basically 1 μS/pixel (275 mS for 512 x 512 picture)

(iv) Status read ($\sim$ 10 μS)

(v) Data transfer from the scratch pad memory $\sim$ 10 mS .

The time taken to transfer data between the host computer and PPP depends on the amount of data: 64 bytes for weight matrix (1 mS), 256 bytes for conversion table (4 mS), and 512 bytes for logical filtering output code or histogram data (10 mS). In the case of loading user's defined microprograms into the program memory of PPP, their loading may be performed typically within 1 mS though

it depends on the number of microprogram instruction words.

Table 4-2 shows a list of execution time of PPP basic picture processing functions for 512 x 512 pictures.

Basic Picture Processing	Execution Time [1]
Two Dimensional Convolution	276 mS
Logical Filtering	285 mS
Data Conversion	279 mS
Histogram Computation	285 mS
Affine Transformation	275 mS
Region Labeling	550 mS [2]
Pixelwise Operations	275 mS [3] ~

1) for 512 x 512 picture ; with refresh cycle time
2) within 254 disjoint regions
3) It depends on the microprogram

Table 4-2 Excecution Time of PPP Basic Picture Processing Functions

From this table, the execution time for the gradient operation for a 512 x 512 picture to calculate the magnitude and direction of gradient vector at each point, which was discussed in Section 4-3, can be estimated as follows:

Main step 1 - two dimensional convolution (x-derivative) 275 mS

Main step 2 - two dimensional convolution (y-derivative) 275 mS

Main step 3 - pixelwise operation for the magnitude calculation 690 mS

Main step 4 - pixelwise operation (division) 415 mS

Main step 5 - data conversion (square root)	278 mS
Main step 6 - pixelwise operation (OR)	275 mS
Total is approximately	2.2 Sec.

A list of execution time of hierachical levels in PPP operations is shown in Table 4-3.

Operation Level	Example	Execution Time
Primitive Instruction	Microprogram Instruction	250nS
	ALU Operation (20-bit)	250nS
	Multiplication (8-bit x 8-bit)	250nS
Primitive Operation	Scratch Pad Memory Access	250nS/Byte
	Picture Memory Access	1 μS/Word
	Sum-of-Products (8x8 Weight Matrix)	1 μS/Pixel
Basic Function	Data Transfer	1mS ~ 10mS
	Picture Processing Function (512x512 Picture)	275mS
	Region Labeling	550mS
Application	Gradient Operation	2.2 Sec

Table 4-3 Execution Time of Hierachical Levels in PPP Operations

Two comparative studies have been accomplished by taking the case of two dimensional convolution. The first of the study compares the convolution by a host computer program and that by PPP. The second part of the comparison is with a general-purpose large-scale computer. The comparison is based on the

total execution time required to complete a given task.

Table 4-4 shows the comparison of the total execution time by PPP, minicomputer, and large-scale computer to perform the two dimensional convolution.

Machine \ Function / Picture Size		Convolution with 8x8		Laplacian (3x3)	
		128x128	512x512	128x128	512x512
PPP		70mS	275mS	70mS	275mS
T-40C		30S	190S	0.6S	72S
TI 5600	GMAP	8.5S	*	0.2S	*
	FORTRAN	50S	*	2.0S	*
	TOSPAX	100S	*	2.0S	*

Table 4-4 Comparison of Total Execution Time for Two Dimensional Convolution

(1) Comparison with Host Minicomputer

The host computer is a TOSBAC-40C minicomputer with 64K bytes core memory. In the first experiment, the size of picture to be processed was determined to be 128x128, such that the host computer may perform a given picture processing task totally in core.

PPP can accomplish the two dimensional convolution with an 8x8 weight matrix for a 128x128 picture data within 70 mS, independent of the weight data. The performance can be improved approximately 18 mS, if a set of line buffers which are at present shift registers is replaced by random access memory. Notice that the system program enforces the width to 512, even if a given

picture has smaller horizontal width than 512, as described in the supplementary remarks in Chapter 3.

The first implementation of the convolution on the host computer entails the complete sum-of-products calculations with 8x8 weights, which results in 64 multiplications, 64 additions and 128 data accesses. A machine language program was written for the TOSBAC-40 host computer to accomplish the two dimensional convolution for a 128x128 picture data. No sophiticated software techniques were undertaken; the program was implemented in a straightforward manner to compute the sum-of-products for each pixel. The program also included the same normalization technique by the bias and shift number as that of PPP.

The two dimensional convolution by TOSBAC-40 is within 30 sec, which is approximately 400 times slower than by PPP as listed in Table 4-4.

A certain set of the two dimensional convolution can often be simplified by replacing the multiplication with shifting, and by neglecting data access and calculation for zero parts of weights.

The second implementation was done for the case of a 3x3 Laplacian operator, one of the most simplest convolutions whose coefficients are 0, 1 and -4. The program was designed to access 5 neighboring pixel data, shift the central pixel value by 2 (equivalently multiply by 4) and add these values (4 additions and 1 subtraction) at each pixel. The normalization of output value was also done so that the program could simulate PPP operation.

The 3x 3 Laplacian operation can be speeded up to 0.6 sec in TOSBAC-40C minicomputer, while PPP needs 70 mS.

The second experiment is for larger size picture processing, for example, 512x512 pixels.

It is assumed that a picture to be processed resides in the disk storage, while the picture memory already contains a picture of interest.

Two kinds of the two dimensional convolution with 8x8 weights and with 3x3 Laplacian weights were performed by TOSBAC-40C and PPP. The time required to transfer picture data between the disk and the core memory of TOSBAC-40C is included in the performance measurement of Table 4-4. Maximum data transfer speed from and to the disk is 800 KB/sec, and average access time is 20 mS.

These experimental results indicate that PPP is a powerful picture processing machine especially for accomplishing the two dimensional convolution. As we compare this result with that of Table 4-3, we notice that the gain of execution speed becomes much more significant when the size of picture is large as well as the size of weight matrix. The picture memory is to be large enough to store whole pictures under processing and thus to reduce the idle time for data transfer between memory hierarchies.

(2) Comparison with Large-Scale Computer

The second comparative study was with a general-purpose large-scale computer (TOSBAC-5600 model 150).

The two dimensional convolution programs with 8x8 weights and 3x3 Laplacian weights for pictures of 128x128 and 512x512 were implemented in three different ways: GMAP machine language, FORTRAN, and TOSPAX. TOSPAX is an extended version of PAX II, developed at the University of Maryland during 1967 - 69, to perform many basic picture processing operations. It consists of a collection of picture processing subroutines which can be called by a FORTRAN program.

TOSBAC-5600 consists of 36-bit words. One word was used to store a pixel data rather than packing four pixels into a word.

The experimental result is also listed in Table 4-4. This comparison indicates that in spite of its ease in writing picture processing programs, TOSPAX results in the slowest implementation.

When we compare PPP with TOSBAC-5600, PPP can accomplish the two dimensional convolution two orders of magnitude faster than large-scale computers.

Table 4-5 lists the comparison of total execution time for some basic picture operations by PPP and TOSBAC-40C: logical filtering, data conversion, affine transformation, and histogram computation. The size of picture was 128x128, such that TOSBAC-40C may perform a picture operation totally in core.

Function \ Machine	PPP	T-40C
Two Dimensional Convolution (8x8)	70mS	30S
Logical Filtering (3x3)	79mS	1S
Data Conversion	22mS	0.2S
Affine Transformation (Rotation)	18mS	0.7S
Histogram Computation	28mS	0.3S

Table 4-5 Comparison of Total Execution Time for Basic Picture Operations (Picture Size - 128x128)

4-5 Applications of PPP Picture Operations

PPP is designed so that it can perform several basic picture operations at high speed. Repetitive computations of pixelwise and local operations, which are frequently used in many applications of picture processing, are implemented in PPP. Specially designed

hardware has improved the picture processing speed by a factor of two orders of magnitude in comparison with conventional digital computers.

PPP picture operations are supervised by the host computer. The host computer transfers a set of parameters and then issues the execution command. Until receiving an end-of-operation interrupt from PPP, the host computer is left free to carry out other tasks in the interim.

This section describes two typical applications of PPP: the contour line processing of a human face and crop identification from remotely sensed data. They include processing of both gray scale pictures and binary pictures, and demonstrate the usage and advantages of PPP.

(1) Contour Detection and Feature Extraction from Human Face

This problem is divided into the following procedures:

(i) Input a digitized picture

(ii) Perform the Laplacian operation

(iii) Detect the contour line by thresholding

(iv) Clean the noise by logical filtering

(v) Extract the feature points by pattern matching.

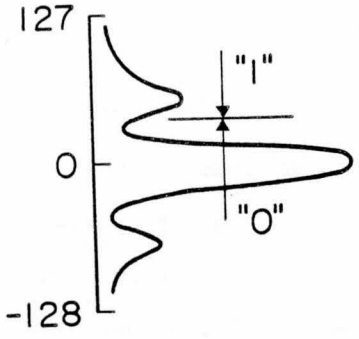
A photograph of a human face is scanned and digitized, and then stored in the picture memory. Fig. 4-9(a) shows a digital picture on the display.

The second-order differentiation is performed by the two dimensional convolution function, whose weight matrix is defined by ∇^2_{w2} in Chapter 2. The output is normalized, such that it may not exceed 8-bit information (-128 ~ +127). (See Fig. 4-9(b).)

Based on the histogram of difference values (Fig. 4-9(c)) which is computed by the histogram computation function, a threshold value is selected either automatically or manually. The data conversion program with the aid of the scratch pad memory

(a) Input Picture

(b) Laplacian Picture
(Absolute Value Display)



(c) Histogram of
Laplacian Values

(d) Thresholded Picture

(e) Noise Cleaned Picture

(f) Feature Point Detection

Figure 4-9 Contour Line Processing of Human Face

converts the Laplacian picture into a binary picture (Fig. 4-9(d)). Then, isolated points and speckle noise are filtered by the logical filter function, incorporating the scratch pad memory (Fig. 4-9(e)).

Several feature points of a human face, which characterize individual persons in a statistical sense, are detected by correlating standard templates. The correlation value with the templates (eye, nose, mouth, chin) is calculated by the two dimensional convolution function, and the points of maximum correlation are located in Fig. 4-9(f). The input picture has several restrictions such that the size of the face is approximately fixed and there is no rotation (i.e., front viewed face).

(2) Crop Identification by Multi-band Photograph

The picture data used in this application study consists of 4 color bands (Blue, Green, Red, and InfraRed) in aerial photographs.

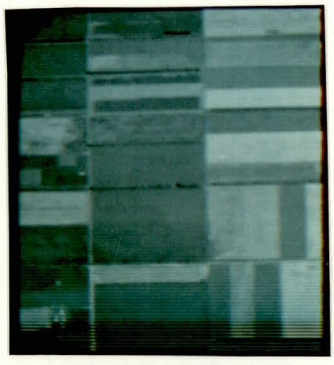
The data was taken by a 4-band camera over an agricultural test site in Hokkaido. The resolution of digitized picture corresponds to 1 m on the ground. Fig. 4-10 is an example of 4-band pictures.

The digitized picture data undergoes the following processings:

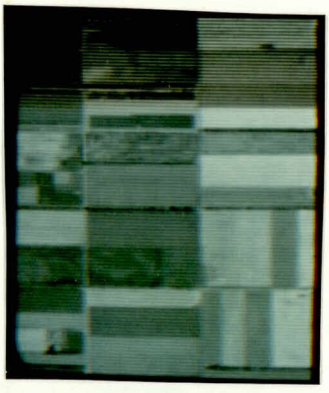
(i)	Preprocessing	Radiometric correction and geometric registration
(ii)	Feature extraction	Statistical parameters calculation and field boundary detection
(iii)	Classification	Crop identification
(iv)	Output	Pseudo-color display.

The radiometric correction is first accomplished to rectify vignetting caused by the optical system in the camera. The pixel-wise addition of the input picture and precalibrated vignetting data is performed by PPP.

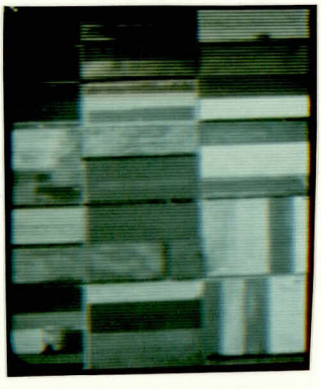
The registration among 4 band pictures requires the affine coordinate transformation. It is assumed that the registration



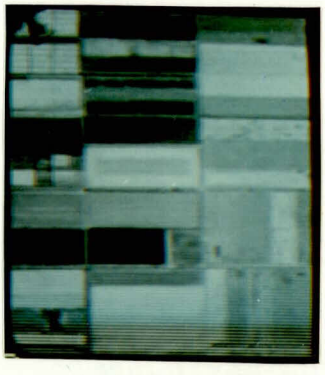
(a) Blue



(b) Green



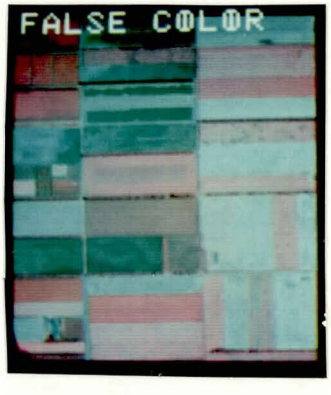
(c) Red



(d) Infra Red

Figure 4 - 10 4 - Band Picture Data

(a) Natural Color
by B, G and R



(b) False Color
by G, R and IR

Figure 4 - 11 Color Composite Pictures after Registration

error is only based on the linear transformation. A set of parameters for the linear coordinate transformation to align each picture (B, G, R) with the IR picture is computed by using their gradient pictures. High speed picture congruencing is afforded by PPP. Fig. 4-11 demonstrates the registration results as two color composite pictures.

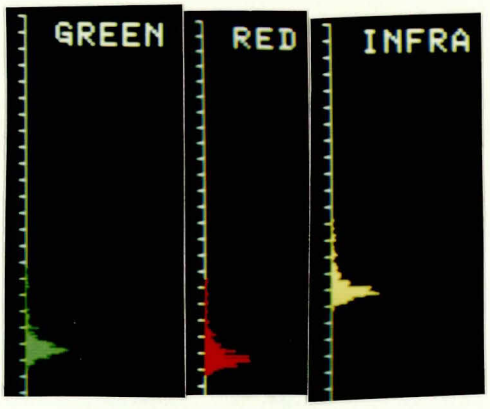
Crop identification is primarily concerned with the classification of crop types in agricultural fields: corn, wheat, soy beans, potatoes etc.

In the first experiment, a very simple method was undertaken to identify crop types, which is based on the following feature extraction and parallel-piped classification techniques. A training area is selected over a crop field of interest automatically or manually as shown in Fig. 4-12(a). The histogram computation function determines the maximum and minimum values in the specified area of each color band (Fig. 4-12(b)). The parallel-piped classification can be executed by the pixelwise ALU operation. The points, whose gray level of each color band is located between the computed maximum and minimum values, are classified as the same class. The result is shown in Fig. 4-12(c).

In the second experiment, a per-field classification technique was proposed to identify the crop type in each field. Field boundaries are first detected by using the two dimensional convolution function to perform the second derivative operation (Laplacian), data conversion function to extract edge-like points by thresholding, and logical filtering function to clean noisy points. (See Figs. 4-13 (a), (b), and (c).)

Then, the Hough transformation is applied to this picture and a final result is given in Fig. 4-13(d). It is assumed that the field boundary consists of only straight lines (i.e., rectangle field). (This Hough transformation was performed in the host computer because of its programming property.)

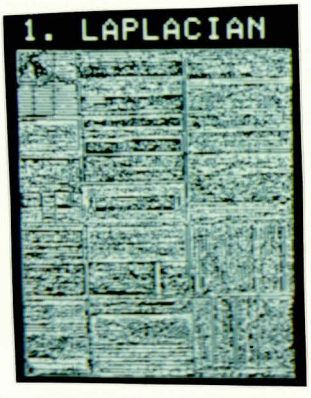
(a) Training Area Set



(b) Histograms

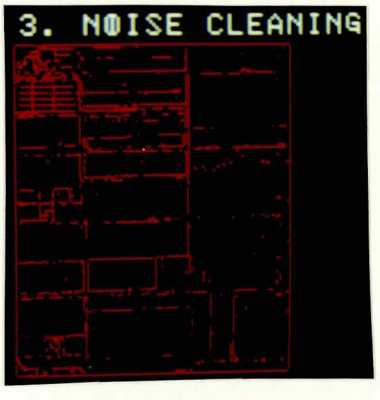
(c) Classification Result

Figure 4-12 Parallel-Piped Classification

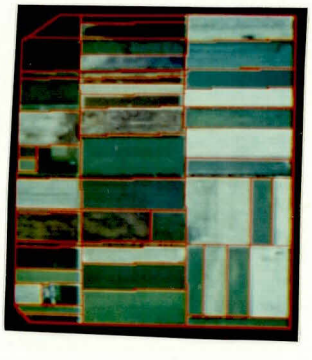


(a) Derivative Operation

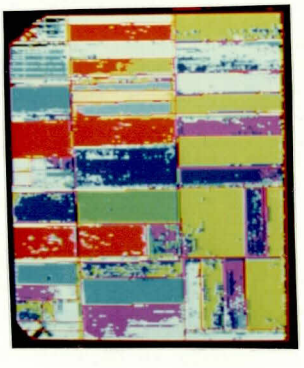
(b) Thresholding



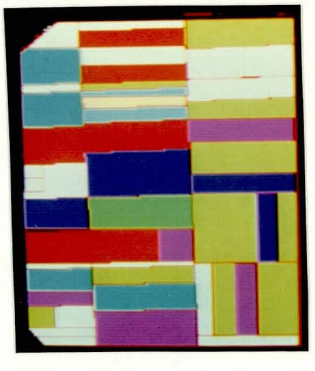
(c) Noise Cleaning



(d) Field Boundary (after Hough Transformation)



(e) Per-Point Classification



(f) Per-Field Classification

Figure 4-13 Per-Field Classification (Crop Identification)

The region labeling function separates this boundary picture into disjoint regions corresponding to agricultural fields, which are used for per-field classification together with per-point classification result (Fig. 4-13(e)). (Per-point classification was also implemented in the host computer.) The majority rule for each labeled field is applied to the per-point classification result so that each field is classified into one of the following crop categories: corn, wheat, soy beans, potatoes, beets, oats, meadow grass, and bare soil. This operation can be implemented by the histogram computation, data conversion, and specially designed microprograms. The final per-field classification result is shown in Fig. 4-13(f).

Other than applications in human-face photograph processing and remotely sensed data analysis, PPP can be applied to the cell detection and analysis, industrial automation, parallax measurement from stereo pairs of aerial photographs etc. The development of application programs in the host computer will greatly emphasize the high speed processing capability and substantial flexibility in PPP hardware and software architectures.

CHAPTER 5 CONCLUSION

This thesis has been devoted to the descriptions of new local parallel picture processing techniques and a newly developed micro-programmable local parallel picture processor.

(1) In Chapter 2, new local parallel picture processing techniques have been applied to edge detection and texture analysis. We have extended the second-order differential operation (the Laplacian) to incorporate the local averaging. The running Laplacian operator has shown good results for human face photographs. A non-linear edge operator was improved by incorporating the maximum derivative value selection and dynamic thresholding. The processing capability was enhanced in the case of natural scenes.

A new concept of planar random walk has been successfully introduced to analyze statistical properties of texture patterns. Texture parameters were calculated by simulating planar random walks in a local area, and used to classify textures.

(2) The careful study of software implementation of local parallel picture processing functions has revealed basic requirements for high speed digital processing hardware. The computations which are most frequently used are selected for the hardware implementation: two dimensional convolution, linear coordinate transformation, logical filtering, data conversion, histogram computation etc.

One of the most commonly used picture processings is found to be the two dimensional convolution, or sum-of-products operation. It requires an intensive use of multiplications, additions, and data accesses. The two dimensional convolution is frequently

used in averaging, differentiation, enhancement, correlation, matching, and so on.

(3) Chapter 3 reports development of the microprogrammable local parallel picture processor (PPP), which can perform basic picture processing functions at high speed, such as the two dimensional convolution, logical filtering, region labeling, linear coordinate transformation, data conversion, histogram computation etc. In order to reduce the hardware cost and complexity to manageable levels, several methods have been explored, which include parallel and pipeline circuit technologies, and a microprogram architecture. The direct connection of PPP with the picture memory has turned out to be substantial to increase capability both for data access and throughput in a picture processing system.

The special-purpose circuits were designed for high speed execution of basic picture processing functions. Pixelwise arithmetic and logic operations can be performed under microprogram control. Rewritable microprogram memory enhances the PPP processing flexibility.

(4) In Chapter 4, software problems were described: the command program in the host computer and the PPP microprogram. The host computer activates the picture memory and PPP, and then issues picture processing commands with a set of parameters. The microprogram controller supervises the internal data flow and the functional behavior of each picture processing module, leaving the host computer free to carry out other tasks in the interim.

Any microprogram defined by a user can be loaded and executed under command program control of the host computer. This capability provides PPP with much flexibility in picture processing operations.

Programming techniques for basic functions were discussed

and their performance was measured. Experimental results have shown that the PPP performance is two orders of magnitude faster than by today's general-purpose computers.

(5) Several picture processing functions have been applied to the contour extraction of human face photographs and crop identification of multiband remotely sensed data. These demonstrations indicate that PPP is sufficiently efficient to justify exploring novel applications and quick analysis in the image understanding studies. Other applications, such as to medical picture analysis, industrial automation, picture manipulation in digital TV equipment, etc. are attainable.

As mentioned by the PPP performance, the hardware realization could not only reduce the processing speed but also lead toward development of new picture processing algorithms, because a high speed processor will make it feasible to accomplish time consuming operations which used to be impracticable in the conventional serial computer. A picture processor is essential in the advanced computer picture analysis system.

144 項欠

REFERENCES

(1) A. Rosenfeld and J. L. Pfaltz, "Sequential Operations in Digital Picture Processing", J. ACM, Vol. 13, No. 4, pp. 471-494, October 1966.

(2) J. L. Pfaltz, J. W. Snively, Jr. and A. Rosenfeld, "Local and Global Picture Processing by Computer", in Pictorial Pattern Recognition, G. C. Cheng, R. S. Ledley, D. K. Pollock and A. Rosenfeld, Eds., Washington, D. C.: Thompson, pp. 353-371, 1968.

(3) M. J. B. Duff, "Parallel Computation in Pattern Recognition", in Methodologies of Pattern Recognition, S. Watanabe, Ed., New York: Academic Press, pp. 133-145, 1969.

(4) C. Mohwinkel and L. Kurz, "Computer Picture Processing and Enhancement by Localized Operations", Computer Graphics and Image Processing, Vol. 5, No. 4, pp. 401-424, 1976.

(5) M. D. Levine, "Feature Extraction: A Survey", Proc. IEEE, Vol. 57, No. 8, pp. 1391-1407, August 1969.

(6) M. Nagao and T. Kanade, "Edge and Line Extraction in Pattern Recognition", Proc. IECEJ, Vol. 55, No. 12, pp. 1618-1627, December 1972 (in Japanese).

(7) L. Davis, "A Survey of Edge Detection Techniques", Computer Graphics and Image Processing, Vol. 4, No. 3, pp. 248-270, 1975.

(8) G. S. Robinson, "Edge Detection by Compass Gradient Masks", Computer Graphics and Image Processing, Vol. 6, No. 4, pp. 492-501, 1977.

(9) R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis, New York: John Wiley and Sons, pp. 271-272, 1973.

(10) G. S. Robinson and J. J. Reis, "A Real-Time Edge Processing Unit", Proc. of the Workshop on Picture Data Description and Management, pp. 155-164, April 1977.

(11) W. Frei and C. C. Chen, "Fast Boundary Detection: A Generalization and a New Algorithm", IEEE Trans. Computers, Vol. C-26, No. 10, pp. 988-998, October 1977.

(12) A. Rosenfeld and M. Thurston, "Edge and Curve Detection for Visual Scene Analysis", IEEE Trans. Computers, Vol. C-20, No. 5, pp. 562-569, May 1971.

(13) C. K. Chow and T. Kaneko, "Automatic Boundary Detection of the Left Ventricle from Cineangiograms", Computers and Biomedical Research, Vol. 5, No. 4, pp. 388-410, August 1972.

(14) M. H. Hueckel, "An Operator which Locates Edges in Digitized Pictures", J. ACM, Vol. 18, No. 1, pp. 113-125, January 1971.

(15) R. Nevatia, "Evaluation of a Simplified Hueckel Edge-Line Detector", Computer Graphics and Image Processing, Vol. 6, No. 4, pp. 582-588, 1977.

(16) Y. Shirai, "Analyzing Intensity Arrays Using Knowledge about Scenes", in The Psychology of Computer Vision, P. H. Winston, Eds., New York: McGraw-Hill, pp. 93-113, 1975.

(17) F. Tomita, Y. Shirai and S. Tsuji, "Texture Analysis", Journal of Information Processing Society of Japan, Vol. 19, No. 2, pp. 173-182, February 1978 (in Japanese).

(18) J. S. Wezka, C. R. Dyer and A. Rosenfeld, "A Comparative Study of Texture Measures for Terrain Classification", IEEE Trans. Syst. Man and Cyber., Vol. SMC-6, No. 4, pp. 269-285, April 1976.

(19) R. M. Haralick, K. Shanmugam and I. Dinstein, "Textural Features for Image Classification", IEEE Trans. Syst. Man and Cyber., Vol. SMC-3, No. 6, pp. 610-621, November 1973.

(20) R. Bajcsy, "Computer Description of Textured Surfaces", Proc. 3rd IJCAI, pp. 572-579, August 1973.

(21) K. Deguchi and I. Morishita, "Texture Analysis by Linear Estimation", Proc. 7th Image Processing Engr. Conf., pp. 13-16, November 1976 (in Japanese).

(22) M. M. Galloway, "Texture Analysis Using Gray Level Run Lengths", Computer Graphics and Image Processing, Vol. 4, No. 2, pp. 172-179, 1975.

(23) H. Tamura, S. Mori and T. Yamawaki, "Psychological and Computational Measurements of Basic Texture Features and Their Comparison", Proc. 3rd IJCPR, pp. 273-277, November 1976.

(24) B. Julesz, "Experiments in the Visual Perception of Texture", Scientific America, Vol.232, No.4, pp.34-43, April 1975.

(25) B. R. Hunt, "Computers and Images", Proc. SPIE/OSA on Image Processing, Vol.74, pp.3-9, August 1976.

(26) A. K. Akers, E. Persoon and K. S. Fu, "A Virtual Memory Computer System for Image Processing", Proc. COMSAC, November 1977.

(27) K. S. Fu, "Special Computer Architectures for Pattern Recognition and Image Processing - An Overview", Proc. AFIPS, Vol.47, pp.1003-1013, June 1978.

(28) K. J. Thurber and L. D. Wald, "Associative and Parallel Processors", Computing Surveys, Vol.7, No.4, pp.215-255, December 1975.

(29) Computing Surveys, Vol.9, No.1, March 1977.

(30) S. H. Unger, "A Computer Oriented Toward Spatial Problems", Proc. IRE, Vol.46, No.10, pp.1744-1750, October 1958.

(31) S. H. Unger, "Pattern Detection and Recognition", Proc. IRE, Vol.47, No.10, pp.1737-1752, October 1959.

(32) B. H. McCormick, "The Illinois Pattern Recognition Computer-ILLIAC III", IEEE Trans. Electron. Comput., Vol.EC-12, No.5, pp.791-813, December 1963.

(33) B. Kruse, "A Parallel Picture Processing Machine", IEEE Trans. Computers,Vol.C-22, No.12, pp.1075-1087, December 1973.

(34) B. Kruse, "The PICAP Picture Processing Laboratory", Proc. 3rd IJCPR, pp.875-881, November 1976.

(35) B. Kruse, "Experience with a Picture Processor in Pattern Recognition Processing", Proc. AFIPS, Vol.47, pp.1015-1024, June 1978.

(36) M. J. B. Duff, B. M. Jones and L. J. Townsend, "Parallel Processing Pattern Recognition System UCPR1", Nuclear Instruments and Methods, Vol.52, No.2, pp.284-288, June 1967.

(37) M. J. B. Duff, "Cellular Logic and its Significance in Pattern Recognition", Proc. AGARD Conf. on Artificial Intelligence, pp.25/1-25/13, 1971.

(38) M. J. B. Duff, D. M. Watson, T. J. Fountain and G. K. Shaw, "A Cellular Logic Array for Image Processing", Pattern Recognition, Vol. 5, No. 3, pp. 229-247, 1973.

(39) M. J. B. Duff, D. M. Watson and E. S. Deutsch, "A Parallel Computer for Array Processing", Proc. IFIP Congress '74, pp. 94-97, 1974.

(40) M. J. B. Duff, "CLIP 4: A Large Scale Integrated Circuit Array Parallel Processor", Proc. 3rd IJCPR, pp. 728-733, November 1976.

(41) M. Yachida, F. Tomita and S. Tsuji, "A High Speed Pattern Processor for Scene Analysis", Trans. IECEJ, Vol. 58-D, No. 4, pp. 208-215, April 1975 (in Japanese).

(42) H. Matsushima, M. Ohyama and Y. Kaido, "Arrayed Processor for Image Processing", Technical Report of Image Processing Research Group of IECEJ, Vol. IE 78-11, pp. 45-53, May 1978 (in Japanese).

(43) H. Yoda, J. Motoike, T. Yasue and M. Ejiri, "Image Processor of Object Recognition", Proc. National Conference of IECEJ, No. 5, pp. 333-334, March 1978 (in Japanese).

(44) G. R. Allen, L. O. Bonrud, J. J. Cosgrove and R. M. Stone, "The Design and Use of Special Purpose Processors for the Machine Processing of Remotely Sensed Data", Proc. LARS Conference on Machine Processing of Remotely Sensed Data, pp. 1A 25-42, October 1973.

(45) P. Juetten, G. Allen, B. Hon and R. Reddy, "An Image Processor Architecture", Proc. Image Understanding Workshop, pp. 12-18, October 1977.

(46) W. J. Bouknight, S. A. Denenberg, D. E. McIntyre, J. M. Randall, A. H. Sameh and D. L. Slotnick, "The ILLIAC IV System", Proc. IEEE, Vol. 60, No. 4, pp. 369-388, April 1972.

(47) K. E. Batcher, "STARAN Parallel Processor System Hardware", Proc. AFIPS, Vol. 43, pp. 405-410, May 1974.

(48) J. M. Voccar and R. O. Faiss, "Warp Processing Using STARAN", Proc. IEEE Workshop on Image Processing and Data Management, pp. 68-76, April 1977.

(49) F. J. Kriegler, et al., "The MIDAS Processor", Proc. 10th ERIM Symposium on Remote Sensing of Environment, pp. 757-768, October 1975.

(50) D. Goodenough, "A Hybrid System for Image Processing and Pattern Recognition in Remote Sensing", IEEE Computer Society Conference on Pattern Recognition and Image Processing, June 1977.

(51) M. J. E. Golay, "Haxagonal Parallel Pattern Transformations", IEEE Trans. Computers, Vol. C-18, No. 8, pp. 733-740, August 1969.

(52) S. B. Gray, "Local Properties of Binary Images in Two Dimensions", IEEE Trans. Computers, Vol. C-20, No. 5, pp. 551-561, May 1971.

(53) D. N. Graham, "Image Transmission by Two-Dimensional Contour Coding", Proc. IEEE, Vol. 55, No. 3, pp. 336-346, March 1967.

(54) J. M. S. Prewitt, "Object Enhancement and Extraction", in Picture Processing and Psychopictorics, B. S. Lipkin and A. Rosenfeld, Eds., New York: Academic Press, pp. 75-149, 1970.

(55) T. Kanade, Computer Recognition of Human Faces, Switzerland: Birkhauser Verlag, 1977.

(56) R. W. Hamming, Numerical Methods for Scientists and Engineers, New York: McGraw-Hill, 1973.

(57) W. Feller, An Introduction to Probability Theory and its Applications, New York: John Wiley & Sons, 1957.

(58) S. Siegel, Non-Parametric Statistics for the Behavioral Sciences, New York: McGraw-Hill, 1956.

(59) P. Brodatz, Textures-A Photographic Album for Artists and Designers, New York: Dover Publications, Inc., 1966.

(60) K. Fukunaga, Introduction to Statistical Pattern Recognition, New York: Academic Press, 1972.

150 項欠

LIST OF PUBLICATIONS AND TECHNICAL REPORTS BY THE AUTHOR

1. "Line Extraction and Pattern Detection in a Photograph"; T. Sakai, M. Nagao, S. Fujibayashi and M. Kidode, Journal of Information Processing Society of Japan, Vol.10, No.3, pp. 132-142, May 1969 (in Japanese).

2. "Processing of Multi-Leveled Pictures by Computer - The Case of Photographs of Human Faces -"; T. Sakai, M. Nagao and M. Kidode, Systems. Computers. Controls, Vol.2, No.3, pp.47-54, 1971 (Originally appeared in Japanese; Trans. IECEJ, Vol.54-C, No.6, pp.445-452).

3. "A New Edge Detection Technique and its Implementation"; H. Wechsler and M. Kidode, IEEE Trans. on Systems, Man, and Cybernetics, Vol.SMC-7, No.12, pp.827-837, December 1977.

4. "A New Edge Detection Technique"; M. Kidode, Technical Report of Processional Group on Image Processing of Information Processing Society of Japan, Vol.IP14-1, pp.1-10, September 1977 (in Japanese).

5. "Texture Analysis by Computer"; M. Kidode and K. S. Fu, Proc. ACM Computer Science Conference, January 1977.

6. "A Random Walk Procedure for Texture Discrimination"; H. Wechsler and M. Kidode, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.PAMI-1, No.3, pp.272-280, July 1979.

7. "Mathematical Models for Scene Analysis"; H. Wechsler and M. Kidode, Proc. 4th IJCPR, pp.755-761, November 1978.

8. "An Interactive System for Image Processing - TOSPICS -"; H. Asada, T. Kondo, H. Numagami, H. Shinoda, M. Kidode and K. Mori, Technical Report of Professional Group on Pattern Recognition and Learning of IECEJ, Vol. PRL75-51, pp.19-30, October 1975 (in Japanese).

9. "Image Processing Software System - TOSPICS EXEC"; K. Mori, M. Kidode, H. Shinoda, H. Asada, H. Numagami and T. Kondo, Proc. 6th Image Processing Engineering Conference, pp.61-64, November 1975 (in Japanese).

10. "Applications of Local Parallel Picture Processor"; H. Asada, H. Shinoda, M. Kidode and K. Mori, Proc. National Conference of IECEJ, p. 155, August 1977 (in Japanese).

11. "An Interactive Image Processing System - TOSPICS - and its Application to Remote Sensing"; H. Shinoda, H. Asada, T. Kondo, K. Mori, M. Kidode and H. Numagami, Proc. 7th AIPR Symp., pp. 177-183, May 1977.

12. "Research and Development of Image Analysis and Recognition System"; K. Mori, Y. Yoneyama, S. Watanabe, H. Shinoda, H. Asada and M. Kidode, Proc. PIPS Seminar, pp. 69-78, July 1977 (in Japanese).

13. "Interactive Image Processing System with Local Parallel Picture Processor"; K. Mori, M. Kidode, H. Shinoda and H. Asada, Proc. National Conference of Electric Engineering, pp. S5/11-14, April 1978 (in Japanese).

14. "Design of Local Parallel Pattern Processor for Image Processing"; K. Mori, M. Kidode, H. Shinoda and H. Asada, Proc. AFIPS, Vol. 47, pp. 1025-1031, June 1978.

15. "Interactive Image Processing System with High-Performance Special Processors"; K. Mori, T. Yoneyama, S. Watanabe H. Shinoda, H. Asada and M. Kidode, Proc. 4th IJCPR, pp. 1125-1129, November 1978.

16. "Hardware Implementation of Image Processing Unit"; M. Kidode, H. Asada, H. Shinoda and S. Watanabe, U.S.-Japan Seminar on Research Towards Real-Time, Parallel Image Analysis and Recognition, November 1978 (originally appeared in Japanese; Technical Report of Professional Group on Image Processing of IECEJ, Vol. IE78-12, pp. 55-65, May 1978).

17. "A Survey of Special Hardware Architectures for High-Speed Image Processing"; M. Kidode and H. Shinoda, Nikkei Electronics, No. 191, pp. 110-140, July 1978 (in Japanese).

18. "Design of Intelligent Image Memory Unit"; M. Tabata, H. Asada, M. Kidode and S. Watanabe, Proc. National Conference of IECEJ, p. 5/7, March 1979 (in Japanese).

19. "An Iterative Prediction and Correction Method for Automatic Stereocomparison"; K. Mori, M. Kidode and H. Asada, Computer Graphics and Image Processing, Vol. 2, No. 3 & 4, pp. 393-401, 1973.

20. "Computer Processing of Remotely Sensed Data"; M. Kidode, H. Shinoda, H. Asada and K. Mori, Technical Report of Professional Group on Pattern Recognition and Learning of IECEJ, Vol. PRL74-11, pp. 31-39, June 1974 (in Japanese).

21. "Crop Identification by Multiband Aerial Photograph"; H. Shinoda, H. Asada, M. Kidode and K. Mori, Proc. 1st Remote Sensing Symp., pp. 17-20, November 1975 (in Japanese).

22. "Interactive Image Processing System and Image Data Retrieval"; M. Kidode, presented at Workshop on Pattern Data Base, December 1977.

23. "Machine Processing of Remotely Sensed Data and Applications"; H. Shinoda, H. Asada, N. Sawada, M. Kidode, K. Mori and S. Watanabe, Proc. ICCS, pp. 861-866, November 1978.

24. "Geometrical Distortion Correction of LANDSAT MSS Images with Local Parallel Picture Processor"; N. Sawada, M. Kidode, H. Shinoda, H. Asada and M. Mino, Proc. Annual Conference of Japan Society of Photogrammetry, pp. 15-18, October, 1978.

25. "Data Base System of LANDSAT MSS Images"; N. Sawada, M. Kidode, H. Numagami, H. Shinoda and T. Kondou, Proc. Annual Conference of JSP, pp. 11 ~ 14, May 1979.

26. "New Image Processing Hardwares and their Applications to Industrial Automation"; H. Asada, M. Tabata, M. Kidode and S. Watanabe, Proc. SPIE Technical Symposium on Imaging Applications for Automated Industrial Inspection and Assembly, Vol. 182, April 1979.

27. "Development of Image Processing Hardwares"; M. Kidode, H. Asada, H. Shinoda and S. Watanabe, Technical Report of Professional Group on Image Display and Processing of Institute of Television Engineering of Japan, Vol. IPD41-3, pp. 31-36, March 1979 (in Japanese).

28. "Image Processing Hardware"; M. Kidode, Proc. Image Processing Symposium, pp. 51-65, June 1979 (in Japanese).

29. "Local Parallel Pattern Processor PPP"; M. Kidode, H. Asada, H. Shinoda, S. Watanabe and K. Mori, Toshiba Review, Vol. 34, No. 6, pp. 511-514, June 1979 (in Japanese).

154 項欠

LIST OF ABBREVIATION

AFN	Affine Transformation Command	110
ALU	Arithmetic and Logic Unit	84
ALUI	ALU Operations with Immediate Data (μC)	118
ALUR	ALU Operations with Registers (μC)	118
ALUX	ALU Operations with Iteration (μC)	118
ASM	Angular Second Moment	56
B	Blue Color Picture	68
BA	512 Bytes (SPM)	107
BL	256 Bytes-Left Half Only (SPM)	107
BR	256 Bytes-Right Half Only (SPM)	107
BSY	Busy Status	105
CALL	Call Subroutine (μC)	116
CD	Central Difference	39
COMP	Comparator	74
CON	Contrast	56
D	Bending Point	91
DCV	Data Conversion Command	111
DR	Data Line for Reading	72
DTR	Data Read from SPM Command	107
DTW	Data Write in SPM Command	107
DU	Device Unavailable Status	106
DW	Data Line for Writing	72
E	End Point	91
EOP	End of Operation Status	105
EPU	Edge Processing Unit	18
EXM	Execute Microprogram Command	112
FFT	Fast Fourier Transform	62
FLT	Spatial Filtering Command	108